DOCUMENT RESUME

ED 143 511 SE 022 988

AUTHOR	Charp, Sylvia; And Others
TITLE	Algorithms, Computation and Mathematics (Fortran Supplement). Teacher's Commentary. Revised Edition.
INSTITUTION	Stanford Univ., Calif. School Mathematics Study Group.
SPONS AGENCY	National Science Foundation, Washington, D.C.
PUB DATE	66
NOTE	104p.; For related documents, see SE 022 983-987; Not available in hard copy due to marginal legibility of original document
EDRS PRICE	MF-$0.83 Plus Postage. HC Not Available from EDRS.
DESCRIPTORS	*Algorithms; *Computers; *Mathematics Education; Programing Languages; Secondary Education; *Secondary School Mathematics; *Teaching Guides
IDENTIFIERS	*FORTRAN; *School Mathematics Study Group

ABSTRACT

This is the teacher's guide and commentary for the SMSG textbook Algorithms, Computation, and Mathematics (Fortran Supplement). The teacher's commentary provides background information for the teacher, suggestions for activities found in the Fortran Supplement, and answers for exercises and activities. The course is designed for high school students in grades 11 and 12. Access to a computer is highly recommended. (RH)

****
* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. Nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *
****

ALGORITHMS, COMPUTATION AND MATHEMATICS

(Fortran Supplement)

Teacher's Commentary

Revised Edition

The following is a list of all those who participated in the preparation of this volume:

Sylvia Charp, Dobbins Technical High School, Philadelphia, Pennsylvania
Alexandra Forsythe, Gunn High School, Palo Alto, California
Bernard A. Galler, University of Michigan, Ann Arbor, Michigan
John G. Herriot, Stanford University, California
Walter Hoffmann, Wayne State University, Detroit, Michigan
Thomas E. Hull, University of Toronto, Toronto, Ontario, Canada
Thomas A. Keenan, University of Rochester, Rochester, New York
Robert E. Monroe, Wayne State University, Detroit, Michigan
Silvio O. Navarro, University of Kentucky, Lexington, Kentucky
Elliott I. Organick, University of Houston, Houston, Texas
Jesse Peckenham, Oakland Unified School District, Oakland, California
George A. Robinson, Argonne National Laboratory, Argonne, Illinois
Phillip M. Sherman, Bell Telephone Laboratories, Murray Hill, New Jersey
Robert E. Smith, Control Data Corporation, St. Paul, Minnesota
Warren Stenberg, University of Minnesota, Minneapolis, Minnesota
Harley Tillitt, U. S. Naval Ordnance Test Station, China Lake, California
Lyneve Waldrop, Newton South High School, Newton, Massachusetts

The following were the principal consultants:

George E. Forsythe, Stanford University, California
Bernard A. Galler, University of Michigan, Ann Arbor, Michigan
Wallace Givens, Argonne National Laboratory, Argonne, Illinois

© 1965 and 1966 by The Board of Trustees of the Leland Stanford Junior University
All rights reserved
Printed in the United States of America

Permission to make verbatim use of material in this book must be secured from the Director of SMSG. Such permission will be granted except in unusual circumstances. Publications incorporating SMSG materials must include both an acknowledgment of the SMSG copyright (Yale University or Stanford University, as the case may be) and a disclaimer of SMSG endorsement. Exclusive license will not be granted save in exceptional circumstances, and then only by specific action of the Advisory Board of SMSG.

Financial support for the School Mathematics Study Group has been provided by the National Science Foundation.

TABLE OF CONTENTS

Chapter		
TF2	INPUT-OUTPUT AND ASSIGNMENT STATEMENTS	
	Summary of Chapter F2	1
	F2-1. Introduction	4
	F2-2. The Character Set	6
	Answers to Exercises.	7
	F2-3. Input-Output Statements.	8
	Answers to Exercises.	10
	Answers to Exercises F2-4	13
	Answers to Exercises F2-6	14
	Answers to Exercises F2-7	15
TF3	BRANCHING AND SUBSCRIPTED VARIABLES	
	Introduction.	19
	Outline of Chapter F3	20
	Answers to Exercises F3-1	22
	Answers to Exercises F3-2	27
	Answers to Exercises F3-3	34
	F3-4. Precedence Levels for Relations.	37
	Answers to Exercises F3-5	38
	Answers to Exercises F3-6	41
TF4	LOOPING	
	Summary of Chapter F4	43
	Answers to Exercises F4-1	45
	Answers to Exercises F4-2	48
	Answers to Exercises F4-3	57
	Answers to Exercises F4-4	58
TF5	SUBPROGRAMS	
	Use of the Computed GO TO Statement	65
	Answers to Exercises F5-1	66
	Answers to Exercises F5-3	67
	Answers to Exercises F5-4	73
	Supplementary Exercises for Section 5-4	79
	Answers to Exercises F5-5	84
	Answers to Exercises F5-6	86
TF7	SOME MATHEMATICAL APPLICATIONS	
	Answers to Exercises F7-1	89
	Answers to Exercises F7-2	92
	Answers to Exercises F7-3	94
	Answers to Exercises F7-4	98
	Answers to Exercises F7-5	99

Chapter TF2

INPUT-OUTPUT AND ASSIGNMENT STATEMENTS

Summary of Chapter F2

Enough of the FORTRAN language is introduced in this chapter to enable a student to write very simple programs. One series of exercises is arranged in such a way as to build up complete programs, any or all of which can be computer tested. These are Exercises 1 - 6 at the end of Sections F2-3 Set A, F2-3 Set C, and F2-7.

In addition Exercise 8, Section F2-7, is also recommended for computer testing. It will be interesting to the better students. You may need to give the students some special help with this one.

The outline

F2-1 Some background on What is FORTRAN and what FORTRAN programs look like.

F2-2 The elements of the language, its characters, numerals for constants and statement labels, variables, names for functions, and operators. An important new idea to be on the lookout for is FORTRAN's distinction between integers and real types for numbers and variables.

F2-3 The READ, PRINT and FORMAT statements are introduced. Format is introduced piecemeal at first; only the simplest concepts of format are discussed (I and F fields and repetition indicators. A-fields are introduced at the end of the chapter). Each succeeding chapter adds more material.

F2-4 The assignment statement is explained largely in terms of what has been learned from the extensive material in the flow chart text. New ideas peculiar to FORTRAN which the student must be on the lookout for are: the notions of prohibition of mixed-mode expressions, and integer division in FORTRAN, which is explained in terms of the greatest integer function.

F2-5 Order of computation in a FORTRAN expression is explained in terms of the material given in the flow chart text.

F2-6 Converting integers to reals and vice versa is shown to be accomplished by the assignment statement.

F2-7 A simple (but complete) FORTRAN program is displayed and a set of exercises given where the student is asked to write his first complete programs.

F2-8 Provides some clerical details in preparing punched card for program statements: the continuation code and the irrelevance of blank spaces.

F2-9 Carriage control for vertical spacing of printed results is introduced. Discussion continues in Chapter 3. This section pertains only to those classes who are using line printers for the output device. With time sharing and remote typewriter consoles becoming available to an increasing extent the topic of carriage control will become unnecessary in a beginning course.

F2-10 Explains how to read into memory alphabetic data in contrast to numeric data. The A-field format code is used. If time is short, this section can be omitted temporarily. It is applied at the end of Chapter 5 in the examples on "string manipulation" and again in Chapter 8.

Literature on FORTRAN II

A non-exhaustive list is provided here.

A. Reference manuals. (The manuals listed are revised frequently and the document numbers that are given may not reflect the latest revision.)

		Computer
1.	"709/7090 FORTRAN" IBM Bulletin C28 - 6054 - 2	IBM 709, 7090
2.	"7070 Series Programming Systems" IBM Bulletin C28 - 6170	7070, 7074
3.	IBM 1620 FORTRAN II specifications IBM Bulletin J26 - 5602 - 1	1620
4.	"FORTRAN for the IBM 1401" IBM Bulletin J24 - 1455 - 0	1401
5.	"FORTRAN for the IBM 1410" IBM Bulletin J24 - 1468 - 0	1410
6.	"FORTRAN System for the Control Data 1604 Computer" Publication 087 A	CDC 1604
7.	"160-A FORTRAN/Reference Manual" Publication 60051302	160 A
8.	"Honeywell 800 Algebraic Compiler Manual" Bulletin DS1 - 44A	HW 800

B. Primers, guides and other texts.

The best source of such material is the SMSG annotated bibliography entitled "Study Guide in Digital Computing and Related Mathematics," which is reprinted at the end of the Teachers Commentary for the Main Text. See especially the references mentioned in Section III, Algorithmic Languages.

F2-1 Introduction

The FORTRAN statement and the declaration

The steps of a FORTRAN program that correspond to the event or action boxes in a flow chart are called statements. To be complete, most FORTRAN programs also include certain statements which are descriptive; these have no direct counterpart in the flow chart language. In Section F2-3, the FORMAT "statement" is re-explained in the new context. It is descriptive or declarative. It is not an event, or action step. At this point we chose to have the student call such nonexecutable statements as declarations rather than statements. This distinction is then maintained in the remainder of the manual.

"Target" programs and source programs

Terminology changes rapidly in a field which is moving as rapidly as the computer field. Most of the older literature used the term object program. We are using the currently preferred term in choosing to use the word target.

In addition to the normal manner of processing a source program as described in the student text, there is an alternative approach which is worth knowing about. In this approach, the processor or compiler program produces a target program which is executed in the "interpretive mode."

This type of target program consists of instructions that are not strictly machine codes. They are machine-like instruction codes, often called an "interpretive" code.

In order to execute such a target a specially developed "interpreter" program must be stored in memory along with the target before execution can proceed. Such compilers have been very successful, especially on machines with limited memory such as the IBM 1620. The interpreter program has the task of interpreting and then carrying out the intent of each pseudo instruction of the target code. The interpreter program in a sense simulates a computer within a computer. The success of these compilers is explained by the fact that a FORTRAN source program translated into interpretive code often occupies far less memory (fewer pseudo instructions) than a FORTRAN source program which is translated into machine code (more actual instructions). In the former (interpretive) case, the total combined memory requirement for the interpretive code produced by the compiler and the interpreter program is normally less than that for the straight machine code. It is for

this reason that this approach is popular for machines of limited memory. On the other hand, the latter (machine code) case normally results in faster-running programs.

Some computers have been designed and built so that the task of compiling is made easy. The Burroughs B-5500 computer is an example. An ALGOL compiler for this computer develops a target program expressed in machine code which is as compact as could be obtained with most interpretive approaches--so the advantages of both storage economy and running speed are thereby achieved with essentially none of the disadvantages.

F2-2 The character set

The characters shown in Table F2-1 are also shown in the card picture, Figure 1-15 in Chapter 1 of the main text.

The special arrangement of letters and digits in Table F2-1 is to emphasize that letters and digits in the same column of the table have one hole punch of their code in common. For example, if you inspect the card picture you will see that the letters E, N, and V and the digit 5 each have a punch in Row 5.

Character	Row punches used
E	12, 5
N	11, 5
V	0, 5
5	5

Special characters

Many of the special characters in Table F2-1 were placed on key punches when FORTRAN became available several years ago. Prior to that time the special characters preferred by the business community were used. The business community was and still is the largest user of key punches. So, if you need key punches for your laboratory classes and obtain the use of a "business" key punch you can expect the keys to display a different set of special characters. There is an equivalence between the FORTRAN set and any other in the sense that up to now all key punch machines of the IBM Model 026 variety punch only one set of hole combinations--i.e., the ones shown on the card picture. Only the characters printed on the keys, and the corresponding characters that print at the very top of the card in each column, may differ. The most typical equivalence is

FORTRAN	Business
(	¢
)	◇ (lozenge)
+	& (ampersand)
=	@ (at, each)
' (apostrophe)	- (dash)

Answers to exercises F2-2 Set A

1. -.02E2
   -.002E3
   -.0002E4

2. Assuming 2.54 cm to the inch, there are $2.54 \times 12 \times 5280$, or 1.609×10^5 cm/mile.

   $1.587 \times 10^{10} \frac{\text{miles}}{\text{lt.yr.}} \times 1.609 \times 10^5 \frac{\text{cm}}{\text{mile}} = 2.553 \times 10^{15} \frac{\text{cm}}{\text{lt.yr.}}$

   In "E" notation this is 2.553E15 .

3. 5.96, 2×10^5, 8.8, -522.4, 5×10^5

4. 3.91E-5, 9.09E0, -6.67E3, 176.4E-9

5. None do.

   1.79×10^{-3} ⇒ 17.9E-4 or 1.79E-3

   6179×10^{-2} ⇒ .6179E-2 or 6179.E-2

   16.79 ⇒ 16.79 or 16.79E0

Statement labels

Although 5 columns are allowed for punching a statement label, oddly enough, not all FORTRAN processors will accept labels as large as 99999. The largest value for a label which is acceptable to all processors is 32768. If the label has less than 5 digits, it may be punched anywhere in the label field (Columns 1 - 5); i.e., blanks are ignored.

Blank Characters in an Expression

In FORTRAN II compilers spaces are ignored when they appear inside an arithmetic expression. Hence spaces appearing within any component of an expression are also ignored. Thus A B L E and ABLE are considered equivalent by the FORTRAN compiler. It happens that there are some other computer languages and their processors, including some versions of FORTRAN IV and PL/I, where blanks are not ignored. In these languages blanks are used as separators between the components of an expression.

It's therefore a good idea to avoid using spaces inside variables or constants. In this way our rules for constructing these components are in harmony with the other languages.

Type of number represented by a variable name

It might be interesting for students to learn that, in more advanced versions of FORTRAN, especially FORTRAN IV, a useful technique known as explicit typing is available to break the leading-letter rule for deciding the variable type. The technique allows us to declare the variable type by a so-called type declaration. For example,

```
INTEGER  T,  P,  Q
REAL     I,  J,  JJ
```

are type declarations which would force the processor to accept T, P, and Q as integer variables and I, J, and JJ as real variables for the program in which these declarations are given.

Many other languages also employ forms of explicit typing.

Answers to Exercises F2-2 Set B

1. 1976S, 19S76, MAR.Y, MAR * Y

2. J97 and N 37 (The space is immaterial.)

F2-3 Input-Output Statements

This is the section where the student is first introduced to the concept of format. FORTRAN format seems to be a necessary evil. No other language has relied as heavily on use of format codes as has FORTRAN. Simplified input-output statements which avoid the explicit use of format codes have, unfortunately, not been available in most FORTRAN implementations. Teachers in the computer field have, for this reason, long complained of this aspect of teaching FORTRAN--but to little avail. On a more positive note we must be quick to point out that great flexibility for controlling the precise forms of data input and desired output is the reward for mastering the use of format code.

We teach only a few of its details at first, in order to focus on the main ideas.

Format code is another language--embedded in the FORTRAN language--and this appears to be the main teaching difficulty. Format language is used to describe information patterns consisting of groups of characters called fields and groups of fields called records, which are normally cards for input or printed lines for output. It is descriptive rather than active. Most FORTRAN statements are imperatives. So format code is supplied in a declaration which is then consulted during the execution of the imperative READ or PRINT statements.

Enlarged pictures of cards and printed lines as visual aids might well be employed here to good advantage in classroom exposition.

E-fields

It is helpful for you to know, but not necessarily for the student to be concerned with at this point, that an alternative form of input and output using "E" notation is available. An E-field code like E15.8 might be used to input and/or output Avogadro's number as shown in Figure TF2-1.

```
      READ  10,  AVGAD
      PRINT 10,  AVGAD
   10 FORMAT (E15.8)
```

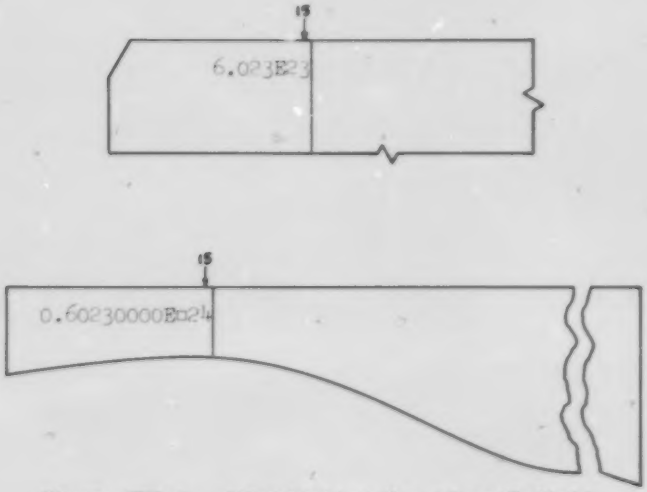


Figure TF2-1. Illustrating Use of an E-field

Exercises F2-3 Set A

Discussion

When assigning these exercises you may wish to suggest that all the students use the same data values because in exercises of Set C of this section the students may want to compute results for later comparison with computer output to be developed with the exercises in Section F2-7. One set of suggested data values is given here.

For Exercise 1 use T = 3.967

" 2 " n = 20, i = 3, and j = 4

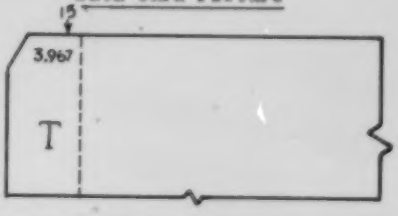
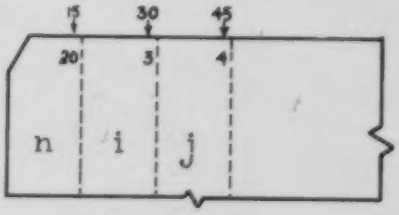
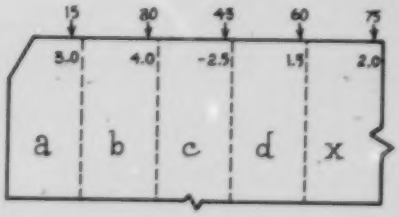
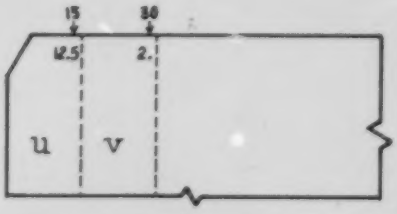
" 3 " a = 3.0, b = 4.0, c = -2.5, d = 1.5, and x = 2.0

" 4 " u = 12.5 and v = 2.0

" 5 " A = 4.14, Y = 2.01

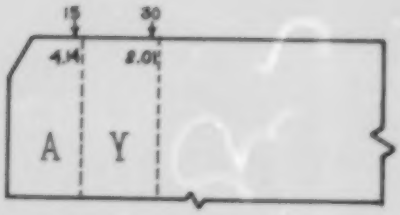
" 6 " r = 10.0, s = 9.0 and PHI = 1.11977 (arcsine of $\frac{9}{10}$)

Answers

	In Box 1	READ and FORMAT	Data Card Picture
1.	T	READ 10, T 10 FORMAT (F15.8)	
2.	n, i, j	READ 11, N, I, J 11 FORMAT (3 I 15)	
3.	a,b,c,d,x	READ 12, A,B,C,D,X FORMAT (5 F15.8)	
4.	u, v	READ 13, U, V 13 FORMAT (2F15.8)	

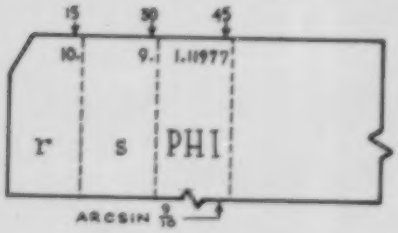
5. A, Y

```
      READ 13, A, Y
13 FORMAT (2 F15.8)
```



6. r,s,PHI

```
      READ 14, R, S, PHI
14 FORMAT (3 F15.8)
```



Answers to exercises F2-3 Set B

In the following answers the format numbers which were chosen are purely arbitrary.

1.
```
      PRINT 51, A,B,J,K,L
51 FORMAT (2F15.8, 3I15)
```
or alternatively, though less desirable,
```
51 FORMAT (F15.8, F15.8, I15, I15, I15)
```

2.
```
      PRINT 52, A,B,C,J,K,L,A1,B1,C1
52 FORMAT (3F15.8, 3I15)
```

3.
```
      PRINT 53, IKE, JAK, BAKER, CHARLY, D,B
53 FORMAT (2I15, 4F15.8)
```

Comment

In Chapter 4 we will expand this subject further and consider some more complicated problems. For example, suppose in Exercise 3, the list contained a seventh item which is real, like T.

Now the format (2I15, 4F15.8) cannot be reused to print the value of T on the second line. Nor may we write (2I15, 5F15.8), because this format would suggest 7 items per line, each 15 columns wide. This corresponds to a line width of 105 spaces. We have, however, agreed to limit line widths to 90 columns in this exercise.

The end-of-record symbol, which is a slash (/), will be introduced in Chapter 4. Its use permits us to write a format to describe two or more printed lines differing in format, or two or more input data cards differing in format.

Thus, we will be able to handle the proposed extension by writing

```
      PRINT 54,  IKE, JAK, BAKER, CHARLY, D, B, T
   54 FORMAT  (2I15, 4F15.8/F15.8)
```

Answers to exercises F2-3 Set C

	In Box 3	PRINT and FORMAT	Appearance of Printed Result (Actual value)
1.	Z	PRINT 20, Z 20 FORMAT (F15.8)	6.46700000
2.	L	PRINT 21, L 21 FORMAT (I15)	44
3.	Z	PRINT 20, Z 20 FORMAT (F15.8)	36.50000000
4.	Q	PRINT 20, Q 20 FORMAT (F15.8)	4.00000000
5.	X	PRINT 20, X 20 FORMAT (F15.8)	0.49114185
6.	AREA	PRINT 20, AREA 20 FORMAT (F15.8)	5.87255xxx

Accuracy to 8 decimal places is not worth requiring of the student. 3 or 4 place accuracy on all results is sufficient. The important thing is

that the student realize that the F15.8 field code will cause the computer to print 8 digits of the result to the right of the decimal point. In the next set of exercises we will ask the student to verify these hand-computed values on the computer.

Answers to exercises F2-4 Set A

1. To express $A^{3/2}$

(a)	ABSF(A ** 1.5)	ABSF function is unnecessary because A is known to be positive so $A^{3/2}$ must also be positive.
(b)	(A ** 3) ** 0.5	O.K., but expensive computationally, (will require log-antilog procedure).
(c)	SQRTF(A ** 3)	This is the best way. (Requires the least amount of computation.)
(d)	ABSF(SQRTF(A ** 3))	ABSF function is unnecessary.
(e)	(A ** 1.5)	O.K., but expensive (requires the log-antilog process).
(f)	ABSF(A) ** 1.5	ABSF function unnecessary. Also, it requires log-antilog process.
(g)	SQRTF(ABSF(A ** 3))	ABSF function unnecessary--otherwise, as good as c.

2. Only c or g would be satisfactory to express $|A|^{3/2}$. g is preferred because it avoids the log-antilog process. Another alternative would be SQRTF(ABSF(A) ** 3).

Answers to exercises F2-4 Set B

In the three incorrect statements below we see that the first and third can be corrected without ambiguity. The second statement may be corrected in one of two ways, but neither can be said to be preferable.

1. T = B* - A should be T = B*(-A)

but

2. F = C/-3 + 4 may be corrected as F = C/(-3) + 4
   or as F = C/(-3 + 4)

   { These in general will yield different results.

Similarly

3. G = A + B*(C* - F/D) may be corrected as
   G = A+B*(C*(-F)/D)
   or G = A+B*(C*(-F/D))

   { These are computationally equivalent.

Answers to exercises F2-6

1. (c) and (d) are invalid (mixed mode).
   (c) should read Y = LOGF(SINF(F)) + 22.
   (d) should read J = J + 1

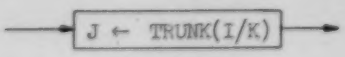
2. A = EXPF(Z ** 2*(A4*Z + A3))
   Avoids log anti-log for powers of Z.
   Factors out Z^2 from subexpression to save 2 multiplications.

3. IPART = V

4. IPART = V
   FTEMP = IPART
   FPART = V - FTEMP

5. RINTV = INTV

6. They are not the same. The box that's needed is



Answers to exercises F2-7

1.

Label	Statement or declaration
1	READ 10, T
10	FORMAT (F15.8)
	Z = 2.5 * T
	PRINT 10, Z
	GO TO 1
	END

2.

Label	Statement or declaration
1	READ 11, N, I, J
11	FORMAT (3I15)
	L = N*(I-1) + J
	PRINT 21, L
21	FORMAT (I15)
	GO TO 1
	END

Note: the PRINT statement may optionally refer to format number 11 which would be equally suitable here, thereby reducing the number of formats needed in this program. If format 11 were used for printing, execution will terminate after printing L, because the list is then exhausted of items in spite of the fact that the governing format would have allowed up to three items to be printed per line. The same comment applies to the answers to the remaining programs in this set.

3.

Label	Statement or declaration
1	READ 12, A, B, C, D, X
12	FORMAT (5F15.8)
	Z = ((A*X + B) * X + C) * X + D
	PRINT 20, Z
20	FORMAT (F15.8)
	GO TO 1
	END

4.

Label	Statement or declaration
1	READ 13, U, V
13	FORMAT (2F15.8)
	Q = SQRTF((U - 4.5)*V)
	PRINT 20, Q
20	FORMAT (F15.8)
	GO TO 1
	END

5.

Label	Statement or declaration
1	READ 13, A, Y
13	FORMAT (2F15.8)
	X = 2./(Y + A/Y)
	PRINT 20, X
20	FORMAT (F15.8)
	GO TO 1
	END

6.

Label		Statement or declaration
1		READ 14, R, S, PHI
14		FORMAT (3F15.8)
		RSQ = R*R
		AREA = 3.14159/2. * RSQ
	1	-(S * SQRTF(RSQ - S*S) + RSQ*PHI)
		PRINT 20, AREA
20		FORMAT (F15.8)
		GO TO 1
		END

The digit 1 in this column is called a "continuation code." When a statement or declaration is too long to fit conveniently on one line it may be continued on succeeding lines. Each of the succeeding lines, if there are more than one must be marked with a non-zero digit. The maximum number of lines permitted for one statement or declaration is not quite standard. Most FORTRAN implementations permit up to 10. You can double check this limit with a reference manual for your situation, although it's not likely to be an important matter.

7. NUM20 = PRICE/2000

8.

Label	Statement or declaration
C	CARNIVAL WHEEL MODEL
C	(LET NS = THE INTEGER S.)
C	(LET NP = THE INTEGER P.)
1	READ 51, NS, M
51	FORMAT (2I15)
	NS = (M + NS) - ((M + NS)/32)*32
C	COMPUTE COLOR POSITION,
C	I.E., (M + NS) MODULO 4
	K = (M + NS) - ((M + NS)/4)*4
C	COMPUTE PAYOFF, NP.
	NP = 20*K - 30
	PRINT 51, NP
	GO TO 1
	END

Chapter TF3

BRANCHING AND SUBSCRIPTED VARIABLES

Introduction

When the student has mastered the flow chart text for Chapter 3 and this companion FORTRAN, he is fully equipped to solve computational problems of essentially any complexity. All the basic programming tools are at his disposal. Much of the material in subsequent chapters falls in the category of important refinements, shorthand techniques, example applications and special concepts. That is why it is very tempting, upon completion of this chapter, to tarry and solve a large number of problems. We avoid this temptation as much as possible, because the refinements to be introduced in each succeeding chapter make the solution of problems increasingly easy and interesting. We expose the student to relatively few applications here. Exercises involving algorithms emphasize analysis mainly rather than synthesis. The shift to synthesis is a gradual process which is accelerated in Chapter 4.

The two fundamental ideas of Chapter 3, branching and subscripted variables are mirrored rather easily in the Chapter F3, requiring few new ideas.

For branching the IF statement is used in FORTRAN. Although its form permits three-way branching, it is used primarily for two-way decision making. The expression being tested is always algebraic rather than relational or logical in value. Consequently the student must learn how to recast a relation like

A < B, true or false

into a form like

A - B, less than zero or not less than zero .

Graded exercises are used to give the student the practice he needs.

The new idea associated with subscripted variables is that the programmer has the responsibility and must declare how much memory space is to be allocated for each vector or array employed in a FORTRAN program. This is done through the DIMENSION declaration.

Outline of Chapter F3

F3-1 The conditional IF statement is introduced. Examples and exercises develop the relationship between flow chart boxes describing the decision or branching action and the FORTRAN equivalent.

Because the IF statement is in the form of a three-way branching tool, and because we want first to learn how to program simple (two-way) condition boxes, the first task is that of learning how the IF statement may be used in two-way decision making.

In the IF statement, expressions are tested not for true or false, but for less than zero, equal to zero or greater than zero. So the student is taught here how to convert or transliterate logical expressions (true or false) into arithmetic expressions whose values may be < 0, $= 0$, or > 0.

At the end of this section another topic in FORMAT is treated. The H-field code is introduced to show how a flow chart box like

can be coded in FORTRAN. H-fields mixed with I- or F-fields in format code permit the output of numerical results to be interspersed with annotation, comments, messages, names, etc. Printer carriage control is shown to be easily accomplished with one-space H-fields like 1Ha, or 1H1.

F3-2 Since the flow chart text covers the subject of auxiliary variables in a way that carries over to FORTRAN in a straightforward fashion, this section consists only of one example program plus exercises.

F3-3 The transliteration of compound condition boxes from flow chart to FORTRAN IF statements is explained and illustrated. Many-way condition boxes can also be transliterated as a series of IF statements. Occasionally the three-way decision power of the IF can be taken advantage of in such transliterations.

F3-4 Since no relational symbols (operators) like

$$<,\ \leq,\ =,\ \neq,\ >,\ \geq$$

are available in FORTRAN II, there is no need to consider the question of precedence levels for such operations.

F3-5 The FORTRAN forms for writing subscripts and singly-subscripted variables are introduced. The DIMENSION declaration for allocating space for one-and two-dimensional arrays is explained and illustrated. The notation for input-output of vector segments is introduced (without explaining the notation as an implied DO-loop).

F3-6 Everything said about singly-subscripted variables in F3-5 is repeated for doubly-subscripted variables. The input-output array segment notation is also introduced (but, again, not explained in terms of DO-loop concepts).

The student is asked to work numerous exercises throughout the chapter which involve construction of FORTRAN programs from corresponding flow charts presented as examples or developed as exercises in the flow chart text.

Answers to Exercises F1-1 Set A

1. Statement 20
2. Statement 30
3. Statement 20
4. =
5. ≠
6. IF(X) 10, 10, 9
7. K - 7 > 0
8. X - 8.4 ≥ 0
9. Y - 4.2 < 0
10. A - B ≤ 0
11. X - Y = 0
12. X + 5 - Y ≠ 0

Answers to Exercises F1-1 Set B

1. IF(W) 5, 8, 8
2. IF(J) 10, 6, 6
3. IF(K - 4) 30, 31, 30
4. IF(X - 9.7) 37, 37, 20
5. IF(I - J) 16, 15, 16
6. IF(A + B - C) 5, 5, 9

Answers to Exercises F1-1 Set C

7.
```
      PRINT 20, X, Y
20    FORMAT(5HbXb=b,F10.3, 5HbYb=b, F10.3)
```

8.
```
      PRINT 21, X
21    FORMAT(1Hb, F10.3, 18HbISbTHEbVALUEbOFbX)
```

9.
```
      PRINT 22, I, J, X
22    FORMAT(1H0,2I6,5HbXb=b, F10.3)
```

10.
```
      PRINT 23
23    FORMAT(31H1FOURbSCOREbANDbSEVENbYEARSbAGO)
```

Answers to Exercises F3-1 Set D

1.

Label	Statement or declaration
	READ 20,B,C,D,X
20	FORMAT (4F15.8)
	PRINT 20,B,C,D,X
	IF (B-C) 5,5,4
4	PRINT 21,D
21	FORMAT (F20.8)
	STOP
5	PRINT 21,X
	STOP
	END

2.

Label	Statement or declaration
	READ 20,B,C,D,X
20	FORMAT (4F15.8)
	PRINT 20,B,C,D,X
	IF (D-C) 4,5,5
4	T=C*B+D*X
	GO TO 6
5	T= D-C
6	PRINT 21,T
21	FORMAT (F20.8)
	STOP
	END

3(a).

Label	Statement or declaration
	READ 20,B,C,D,X
	PRINT 20,B,C,D,X
20	FORMAT (4F15.8)
	T=B+C
	U=B*C*D
	IF (T*T+X*X-U) 7,7,5
5	W=T+U
	PRINT 21,W
	STOP
7	W=U*U
	Y=T**8
	PRINT 21,W,Y
	STOP
21	FORMAT (2F20.8)
	END

3(b). Label Statement or declaration

```
      READ 20,B,C,D,X
      PRINT 20,B,C,D,X
   20 FORMAT (4F15.3)
      IF ((B+C)**2+X*X-B*C*D) 7,7,5
    5 W=C+B+P*C*D
      PRINT 21,W
      STOP
    7 W=B*B*C*C*D*D
      Y=(B+C)**8
      PRINT 21,W,Y
      STOP
   21 FORMAT (2F20.8)
      END
```

4. Label Statement or declaration

```
    1 READ 15,J,M,N
   15 FORMAT (3I15,F20.8)
      IF (M-N) 4,4,3
    3 SUM = J+M
      GO TO 5
    4 SUM= J+N
    5 PRINT 15,J,M,N,SUM
      GO TO 1
      END
```

5. Label Statement or declaration

```
    1 READ 25,B,C
      PRINT 25,B,C
   25 FORMAT (2F15.8)
      IF (B) 5,4,5
    5 X=-C/B
      PRINT 26,X
   26 FORMAT (24H THE ROOT OF B*X+C=0 IS F15.8)
      GO TO 1
    4 IF (C) 7,6,7
    7 PRINT 27
C     STATEMENTS 6 AND 7 HAVE EMPTY OUTPUT LISTS
   27 FORMAT (20H B*X+C=0 HAS NO ROOT)
      GO TO 1
    6 PRINT 28
   28 FORMAT (36H EVERY REAL NUMBER SATISFIES B*X+C=0)
      GO TO 1
      END
```

Answers to Exercises F3-1 Set E

1.

Label	Statement or declaration
	SUMALL = 0
	NCOUNT = 1
2	READ 20, T
20	FORMAT(F10.5)
	SUMALL = SUMALL + T
	NCOUNT = NCOUNT + 1
	IF(NCOUNT - 100) 2, 2, 6
6	PRINT 21, SUMALL
21	FORMAT (10H□SUMALL□=□, F12.5)
	STOP
	END

2.

Label	Statement or declaration
	SUMCUB = 0
	NCOUNT = 1
2	READ 20, T
20	FORMAT(F10.5)
	SUMCUB = SUMCUB + T**3
	NCOUNT = NCOUNT + 1
	IF(NCOUNT - 100) 2, 2, 6
6	PRINT 21, SUMCUB
21	FORMAT(10H□SUMCUB□=□, F16.5)
	STOP
	END

3.

Label	Statement or declaration
	SUMNEG = 0
	NCOUNT = 1
2	READ 20, T
20	FORMAT (F10.5)
	IF(T) 4, 5, 5
4	SUMNEG = SUMNEG + T
5	NCOUNT = NCOUNT + 1
	IF(NCOUNT - 100) 2, 2, 7
7	PRINT 21, SUMNEG
21	FORMAT(10H□SUMNEG□=□, F12.5)
	STOP
	END

4.

Label	Statement or declaration
	SUMALL = 0
	SUMCUB = 0
	SUMNEG = 0
	NCOUNT = 1
2	READ 20, T
20	FORMAT(F10.5)
	SUMALL = SUMALL + T
	SUMCUB = SUMCUB + T**3
	IF(T) 5, 6, 6
5	SUMNEG = SUMNEG + T
6	NCOUNT = NCOUNT + 1
	IF(NCOUNT - 100) 2, 2, 8
8	PRINT 21, SUMALL, SUMCUB, SUMNEG
21	FORMAT(10H□SUMALL□=□, F12.5,
1	10H□SUMCUB□=□, F16.5,
2	10H□SUMNEG□=□, F12.5)
	STOP
	END

5.

Label	Statement or declaration
	CUMSUM = 0
	NCOUNT = 1
2	READ 20, T
20	FORMAT(F10.5)
	CUMSUM = CUMSUM + T
	PRINT 21, CUMSUM
21	FORMAT(18H□CUMULATIVE□SUM□=□, F12.5)
	NCOUNT = NCOUNT + 1
	IF (NCOUNT - 100) 2, 2, 7
7	STOP
	END

6.

Label	Statement or Declaration
C	BADMINTON OR VOLLEYBALL
	I = 0
	ISTATE = 0
	ISCA = 0
	ISCB = 0
2	IF(I - 100) 3, 12, 3
3	READ 101, T
	IF(T) 6, 5, 5
5	IF(ISTATE) 8, 7, 8
7	ISCA = ISCA + 1
11	I = I + 1
	GO TO 2
8	ISTATE = 0
	GO TO 11
6	IF(ISTATE) 10, 9, 10
9	ISTATE = 1
	GO TO 11
10	ISCB = ISCB + 1
	GO TO 11
12	IF(ISCA - ISCB) 14, 13, 14
13	PRINT 102, SCA
	STOP
14	IF(ISCA - ISCB) 16, 16, 15
15	PRINT 103, SCA, SCB
	STOP
16	PRINT 104, SCB, SCA
	STOP
101	FORMAT(F12.5)
102	FORMAT(10H□TIE□GAME□,I6,4H□ALL)
103	FORMAT(14HPLAYER□A□WINS□,I6,4H□TO□,I6)
104	FORMAT(14HPLAYER□B□WINS□,I6,4H□TO□,I6)
	END

Answers to Exercises F3-2 Set A

1.

Label	Statement or declaration
C	FIBONACCI SEQUENCE USED TO PRODUCE RANDOM NUMBERS
	LTERM = 1
	NLT = 0
	I = 1
2	IF(I - 117) 3, 6, 6
3	ICOPY = LTERM
C	LOPPING OFF ALL BUT RIGHTMOST THREE DIGITS
	LTERM = LTERM + NLT - 1000 × ((LTERM + NLT)/1000)
	NLT = ICOPY
	I = I + 1
	IF(I - 17) 3, 4, 4
4	PRINT 101, I, LTERM
	GO TO 2
6	STOP
101	FORMAT(2I6)
	END

2.

Label	Statement or declaration
C	THE TWOSUM PROBLEM
	READ 100, TOLD
	I = 2
5	READ 100, TNEW
	TWOSUM = TNEW + TOLD
	PRINT 101, TWOSUM
	TOLD = TNEW
	I = I + 1
	IF(I - 100) 3, 3, 8
8	STOP
100	FORMAT(F15.8)
101	FORMAT(10H␣TWOSUM␣=␣,F15.8)
	END

3.

Label	Statement or declaration
C	THE ALTSUM PROBLEM
	READ 100, TOLDER
	READ 100, TOLD
	I = 3
4	READ 100, TNEW
	ALTSUM = TOLDER + TNEW
	PRINT 101, ALTSUM
	TOLDER = TOLD
	TOLD = TNEW
	I = I + 1
	IF(I - 100) 4, 4, 9
9	STOP
100	FORMAT(F15.8)
101	FORMAT(10H␣ALTSUM␣=␣,F15.8)
	END

4.

Label	Statement or declaration
C	MOVING AVERAGE
	READ 100, K
	READ 101, TOLDER
	READ 101, TOLD
	I = 3
5	READ 101, TNEW
6	IF(I - K) 7, 8, 8
7	TOLDER = TOLD
	TOLD = TNEW
	I = I + 1
	GO TO 5
8	IF(TNEW - TOLD) 9, 10, 10
9	AVERGE = 0.5 * (TOLD + TOLDER)
	GO TO 11
10	AVERGE = (TOLDER + TOLD + TNEW)/3.
11	PRINT 102, AVERGE
	IF(I - 100) 7, 13, 13
13	STOP
100	FORMAT(I6)
101	FORMAT(F15.8)
102	FORMAT(12H␣AVERAGE␣IS␣,F15.8)
	END

5.

Label	Statement or declaration
C	THE REGIONS TABLE
	PRINT 101
	N = 0
	IA = 1
	ISUMA = 1
	IB = 1
	ISUMB = 1
	IC = 1
3	PRINT 102, N, IA, IB, IC
	IF(N - 15) 5, 5, 6
5	IC = ISUMB + 1
	IB = ISUMA + 1
	IA = IA + 1
	N = N + 1
	ISUMB = ISUMB + IB
	ISUMA = ISUMA + IA
	GO TO 3
6	STOP
101	FORMAT(24H□□□□□N□□□□A□□□□□B□□□□□C)
102	FORMAT(4I6)
	END

Answer to Exercise F3-2 Set B

Label	Statement or declaration
	READ 100, KC, KD
	KA = KC
	KB = KD
	IF(KA - KB) 5, 5, 4
4	KR = KA
	GO TO 11
5	IF(KA) 6, 7, 6
6	KR = KB - KB/KA*KA
11	KB = KA
	KA = KR
	GO TO 5
7	IF(KB) 9, 8, 9
8	KX = 0
	GO TO 10
9	KX = KC*KD/KB
10	PRINT 101, KC, KD, KX
	STOP
100	FORMAT(2I6)
101	FORMAT(1H□,3I6)
	END

Answers to Exercises F3-2 Set C

1.

Label	Statement or declaration
	READ 20,X1,Y1,X2,Y2
	PRINT 20,X1,Y1,X2,Y2
20	FORMAT (4F15.3)
	DIST= SQRTF((X2-X1)**2+(Y2-Y1)**2)
	PRINT 21,DIST
21	FORMAT (21H THE LENGTH OF PQ IS F15.3)
	GO TO 1
	END

2.

Label	Statement or declaration
1	READ 20,X1,Y1,X2,Y2
	PRINT 20,X1,Y1,X2,Y2
20	FORMAT (4F15.3)
3	IF (X2-X1) 4,6,4
4	S=(Y2-Y1)/(X2-X1)
	PRINT 21,S
21	FORMAT (20H THE SLOPE OF PQ IS F20.3)
	GO TO 1
6	PRINT 22
22	FORMAT (29H PQ IS PARALLEL TO THE Y-AXIS)
	GO TO 1
	END

3.

Label	Statement or declaration
1	READ 20,X1,Y1,X2,Y2
	PRINT 20,X1,Y1,X2,Y2
20	FORMAT (4F15.3)
	READ 20,DELX
	PRINT 24, DELX
24	FORMAT(8H DELX =F15.3)
	IF (X2-X1) 5,8,5
5	S= (Y2-Y1)/(X2-X1)
	DELY=S*DELX
	PRINT 21,DELY
21	FORMAT (8H DELY = F15.3)
	GO TO 1
8	IF (DELX) 9,10,9
9	PRINT 22
22	FORMAT (21H NO SUCH VALUE EXISTS)
	GO TO 1
10	PRINT 23
23	FORMAT (24H ANY REAL NUMBER WILL DO)
	GO TO 1
	END

4. Label Statement or declaration

```
      1 READ 20,X1,Y1,X2,Y2
        PRINT 20,X1,Y1,X2,Y2
     20 FORMAT (4F15.3)
        READ 20,DELY
        PRINT 24, DELY
     24 FORMAT(8H DELY= F15.3)
        IF (Y1-Y2) 5,10,5
      5 IF (X1-X2) 6,9,6
      6 S= (Y2-Y1)/(X2-X1)
        DELX= DELY/S
      8 PRINT 21,DELX
     21 FORMAT (8H DELX = F15.3)
        GO TO 1
      9 DELX=0
        GO TO 8
     10 IF (DELY) 11,12,11
     12 PRINT 23
     23 FORMAT (24H ANY REAL NUMBER WILL DO)
        GO TO 1
     11 PRINT 22
     22 FORMAT (21H NO SUCH VALUE EXISTS)
        GO TO 1
        END
```

5. Label Statement or declaration

```
      1 READ 20,X1,Y1,X2,Y2
        PRINT 20,X1,Y1,X2,Y2
     20 FORMAT (4F15.3)
        READ 20,X
        PRINT 24, X
     24 FORMAT(5H X= F15.3)
        IF (X1-X2) 5,8,5
      5 S= (Y2-Y1)/(X2-X1)
        Y= Y1+S*(X-X1)
        PRINT 21,Y
     21 FORMAT (5H Y = F15.3)
        GO TO 1
      8 IF (X-X1) 9,10,9
      9 PRINT 22
     22 FORMAT (21H NO SUCH VALUE EXISTS)
        GO TO 1
        PRINT 23
     23 FORMAT (24H ANY REAL NUMBER WILL DO)
        GO TO 1
        END
```

6.

```
Label   Statement or declaration
    1   READ 20,X1,Y1,X2,Y2
        PRINT 20,X1,Y1,X2,Y2
   20   FORMAT (4F15.3)
        READ 20,Y
        PRINT 24, Y
   24   FORMAT(5HOYD=0F15.3)
        IF (Y1-Y2) 5,10,5
    5   IF (X1-X2) 6,9,6
    6   S=(Y2-Y1)/(X2-X1)
        X= X1+ (Y-Y1)/S
    8   PRINT 21,X
   21   FORMAT (5H X =  F15.3)
        GO TO 1
    9   X=X1
        GO TO 8
   10   IF (Y-Y1) 11,12,11
   11   PRINT 22
   22   FORMAT (21H NO SUCH VALUE EXISTS)
        GO TO 1
   12   PRINT 23
   23   FORMAT (24H ANY REAL NUMBER WILL DO)
        GO TO 1
        END
```

7.

```
Label   Statement or declaration
    1   READ 20,X1,Y1,X2,Y2
        PRINT 20,X1,Y1,X2,Y2
   20   FORMAT (4F15.3)
        IF (X1-X2) 3,11,3
    3   IF (Y1-Y2) 4,9,4
    9   PRINT 21
        PRINT 22,Y1
        GO TO 1
    4   S=(Y2-Y1)/(X2-X1)
        XINT= X1-Y1/S
        YINT=-S*XINT
        PRINT 23,XINT
        PRINT 22,YINT
        GO TO 1
   11   PRINT 23,X1
        PRINT 24
        GO TO 1
   21   FORMAT (29H PQ DOES NOT INTERSECT X-AXIS)
   22   FORMAT (16H Y-INTERCEPT IS F15.3)
   23   FORMAT (16H X-INTERCEPT IS F15.3)
   24   FORMAT (29H PQ DOES NOT INTERSECT Y-AXIS)
        END
```

8.

Label	Statement or declaration
1	READ 20,X1,Y1,X2,Y2
	PRINT 20,X1,Y1,X2,Y2
20	FORMAT (4F15.3)
	IF (X1-X2) 3,11,3
3	S = (Y2-Y1)/(X2-X1)
	XINT = X1-Y1/S
	YINT = Y1-S*X1
6	IF (Y1*Y2) 7,10,10
10	PRINT 21
	GO TO 8
7	PRINT 23, XINT
8	IF (X1*X2) 9,12,12
9	PRINT 22, YINT
	GO TO 1
12	PRINT 24
	GO TO 1
11	XINT = X1
	GO TO 6
21	FORMAT (29H PQ DOES NOT INTERSECT X-AXIS)
22	FORMAT (16H Y-INTERCEPT IS F15.3)
23	FORMAT (16H X-INTERCEPT IS F15.3)
24	FORMAT (29H PQ DOES NOT INTERSECT Y-AXIS)
	END

Answers to Exercises F3-3 Set A

1. Label | Statement or declaration

```
    | IF (2.0-X) 2,2,30
  2 | IF (X-7.0) 20,20,30
```

2. Label | Statement or declaration

```
    | IF (7.0-Q) 20,2,2
  2 | IF (7.0-R) 20,3,3
  3 | IF (7.0-S) 20,30,30
```

3. Label | Statement or declaration

```
    | IF (1.7-X) 2,30,30
  2 | IF (X-8.4) 3,30,30
  3 | IF (-3.9-Y) 4,30,30
  4 | IF (Y-5.4) 20,30,30
```

4(a) Label | Statement or declaration

```
    | IF (X1) 30,30,2
  2 | IF (0.5*X1-Y1) 3,30,30
  3 | IF (Y1-2.0*X1) 20,30,30
```

4(b) Label | Statement or declaration

```
    | IF (X1) 5,30,30
  5 | IF (2.0*X1-Y1) 6,30,30
  6 | IF (Y1-0.5*X1) 20,30,30
```

4(c) Label | Statement or declaration

```
    | IF (X1) 5,30,2
  2 | IF (0.5*X1-Y1) 3,30,30
  3 | IF (Y1-2.0*X1) 20,30,30
  5 | IF (2.0*X1-Y1) 6,30,30
  6 | IF (Y1-0.5*X1) 20,30,30
```

5. Label | Statement or declaration

```
    | IF (X1) 30,30,2
  2 | IF (Y1) 30,30,3
  3 | IF (Y1-(-0.6667*X1+2.0)) 20,30,30
```

6. Label | Statement or declaration

```
    | IF (X1-3.14159) 2,2,30
  2 | IF (0.5*(3.14159-X1)-Y1) 3,3,30
  3 | IF (Y1-SINF(X1)) 20,20,30
```

7. Label | Statement or declaration

```
    | IF (Y1) 30,30,2
  2 | IF (Y1 - (-4.0 * X1 + 16.0))30,3,3
  3 | IF (Y1 - (4.0 * X1 - 12.0))30,20,20
```

Answers to Exercises F3-3 Set B

1.
```
      IF(I - 7) 2, 10, 10
10    IF(I - 20) 3, 3, 4
```

2.
```
      IF(W - A) 1, 10, 10
10    IF(W - (B - 2)) 2, 11, 11
11    IF(W - C) 3, 3, 4
```

3.

Label	Statement or declaration
1	READ 20,X1,Y1,X2,Y2
	PRINT 20,X1,Y1,X2,Y2
	IF (X1) 10,4,7
7	IF(Y1) 9,6,8
8	PRINT 21
	GO TO 1
6	PRINT 26
	GO TO 1
9	PRINT 22
	GO TO 1
4	PRINT 25
	IF (Y1) 1,13,1
13	PRINT 27
	GO TO 1
10	IF (Y1) 12,6,11
11	PRINT 23
	GO TO 1
12	PRINT 24
	GO TO 1
20	FORMAT (4F15.3)
21	FORMAT (2H 1)
22	FORMAT (2H 4)
23	FORMAT (2H 2)
24	FORMAT (2H 3)
25	FORMAT (21H P LIES ON THE Y-AXIS)
26	FORMAT (21H P LIES ON THE X-AXIS)
27	FORMAT (35H P IS THE ORIGIN)
	END

4. Label Statement or declaration

```
    C   CARNIVAL WHEEL MODEL
      1 READ 51,NS,M
     51 FORMAT (2I5)
        K=M+NS - ((M+NS)/4)*4
     40 IF (K) 41,5,41
     41 IF(K-1) 42,6,42
     42 IF (K-2) 7,7,8
      5 NP=-20
        GO TO 9
      6 NP= -30
        GO TO 9
      7 NP=0
        GO TO 9
      8 NP=50
      9 PRINT 51,NP
        GO TO 1
        END
```

Comment: For more efficiency we could use just two IF statements. Statement 40 would be removed. Statements 41 and 42 would be

```
41   IF(K - 1) 5, 6, 42
42   IF(K - 2) 7, 7, 8
```

We use the first IF as a three-way branch here.

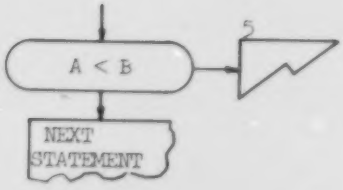
F3-4 Precedence Levels for Relations

In more advanced versions of FORTRAN, especially FORTRAN IV or variations of it, relational operators can be used in an IF statement called the "logical IF" as distinguished from the one discussed in this chapter which is usually called the "arithmetic IF."

In the logical IF one could write for example

```
IF (A .LT. B) GO TO 5
```

which is equivalent to



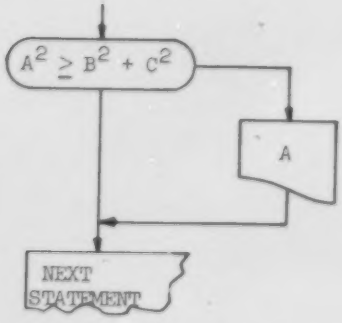
Here the character group .LT. means "is less than." Other character groupings are used to designate the five other relational operator symbols--all of which are listed here.

$<$	$\leq$	$=$	$\neq$	$>$	$\geq$
.LT.	.LE.	.EQ.	.NE.	.GT.	.GE.

As another example, one could write

```
IF(A**2 .GE. B**2 + C**2)PRINT 5, A
```

which is equivalent to



Precedence levels for these relational operators are exactly as described in Section 3-4.

Answers to Exercises F3-5 Set A

1. X(5)
2. X(J)
3. CHAR(I) use upper case for letter i.
4. B(I + 2)
5. X cannot be used as a subscript since it is not an integer variable.
6. Z(5*I + 2)
7. I + J is an invalid form for a subscript expression (variable 1 + variable 2 not allowed).
8. -4 not valid; only unsigned integer constants (non-zero) are permitted.
9. XYZ(IXY)
10. -5*J would be an invalid subscript expression. An unsigned integer constant (non-zero) is called for in this.

Answers to Exercises F3-5 Set B

1.
```
DIMENSION A(50)
READ 5, (A(I), I = 1, K)
```

2.
```
DIMENSION B(125)
READ 6, (B(J), J = 5, N, 2)
```

3.
```
DIMENSION A(50), B(50)
READ (A(I), I = 1, N, 10), (B(I), I = 10, N, 2))
```

Answers to Exercises F3-5 Set C

1.

Label	Statement or declaration
C	CARNIVAL WHEEL WITH SUBSCRIPTS
	DIMENSION NP(4)
	READ 51,(NP(I),I=1,4)
1	READ 51,NS,M
51	FORMAT(2I15)
	K=M+NS - ((M+NS)/4) * 4
	PRINT 51,NP(K+1)
	GO TO 1
	END

2.

Label	Statement or declaration
C	FIGURE 3-29
	DIMENSION B(100)
	I=1
	IANY=0
	READ 20,C
20	FORMAT(F10.5)
	READ 21,(B(I),I=1,100)
21	FORMAT(7F10.5)
4	IF (B(I)-C) 5,8,8
8	IANY=1
	PRINT 22,I,B(I)
22	FORMAT(I5,F10.5)
5	I=I+1
	IF (I-100) 4,4,7
7	IF (IANY) 11,10,11
10	PRINT 23
23	FORMAT (5H NONE)
11	STOP
	END

3.

Label	Statement or declaration
C	FINDING THE ACTUAL DEGREE OF A POLYNOMIAL
	DIMENSION A(51)
1	READ 100, N
	NN = N + 1
C	NN IS FORTRAN UPPER BOUND (ONE HIGHER)
	READ 101,(A(I), I = 1, NN)
3	IF(A(NN)) 6, 4, 6
4	NN = NN - 1
C	LOWER BOUND IS ONE HIGHER ALSO
	IF (NN - 1) 7, 3, 3
7	PRINT 102
	GO TO 1
6	N = NN - 1
	PRINT 103, N
	PRINT 104, (A(I), I = 1, NN)
	GO TO 1
100	FORMAT (I5)
101	FORMAT(5F10.5)
102	FORMAT(20H□POLYNOMIAL□IS□EMPTY)
103	FORMAT(22H□POLYNOMIAL□DEGREE□IS□,I5,
1	31H.□□COEFFICIENTS□ARE□AS□FOLLOWS.)
104	FORMAT(1H□,5F15.5)
	END

Answers to Exercise F3-5 Set D

Label	Statement or declaration
C	ORCHESTRAVILLE
	DIMENSION IA(500)
1	READ 100, N, (IA(K),K = 1,N)
100	FORMAT(10I5)
C	FIRST DATA CARD HAS VALUE OF N IN THE FIRST FIELD,
C	THEN UP TO 9 AGES.
3	K = 1
4	IF(IA(K) - IA(K+L)) 6, 6, 5
6	K = K+1
	IF(K - N) 4, 8, 8
5	ICOPY = IA(K)
	IA(K) = IA(K+1)
	IA(K+1) = ICOPY
	GO TO 3
8	MID = N/2
C	TEST N TO SEE IF IT IS EVEN FOR COMPUTING MAD
	IF(MID*2 - N) 11, 10, 11
10	MAD = MID + 1
	GO TO 12
11	MAD = MID
12	SUM = IA(MID) + IA(MAD)
	TMEED = .5 * SUM
	PRINT 101, TMEED, IA(1),IA(N)
101	FORMAT(15H□MEDIAN□AGE□IS□I5,
1	13H□YOUNGEST□IS□,I5,11H□OLDEST□IS□,I5)
	GO TO 1

Note that several names of variables must be changed from those used in the flow chart in order to designate the desired integer or floating point modes.

Answers to Exercises F3-6

1.

Label	Statement or declaration
	DIMENSION P(22, 27)
	COLSUM = 0
	I = 1
3	IF(I - 12) 4, 5, 4
4	COLSUM = COLSUM + P(I, K)
5	IF(I - 22) 6, 7, 7
6	I = I + 1
	GO TO 3
7	PRINT 50, COLSUM

2.

Label	Statement or declaration
	DIMENSION P(22, 27)
	J = 1
2	P(L, J) = P(L, J) + P(M,J)
	IF(J - 27) 4, 5, 5
4	J = J + 1
	GO TO 2

3.

Label	Statement or declaration
	DIMENSION P(22, 27)
	J = 1
2	IF(J - K) 3, 4, 3
3	P(L, J) = P(L, J) + 2.0 × P(M, J)
4	IF(J - 27) 5, 6, 6
5	J = J + 1
	GO TO 2

4.

Label	Statement or declaration
	DIMENSION P(22,27)
	J = 1
2	COPY = P(L,J)
	P(L,J) = P(M,J)
	P(M,J) = COPY
	IF(J - 27) 4, 5, 5
4	J = J + 1
	GO TO 2

5.

Label	Statement or declaration
	DIMENSION P(22,27)
	I = 1
	MAX = 0
3	IF(ABSF(P(L,J)) - ABSF(MAX))5,5,4
4	MAX = P(L,J)
5	IF(J - 27) 6, 7, 7
6	J = J + 1
	GO TO 3
7	J = 1
8	P(L,J) = P(L,J)/MAX
	IF(J - 27) 10, 11, 11
10	J = J + 1
	GO TO 8

Chapter TF4

LOOPING

The main sections of this chapter are:

F4-1 The DO statement
F4-2 Illustrative examples
F4-3 Table-look-up
F4-4 Nested DO loops

This chapter follows closely Chapter 4 of the Flow Chart text.

F4-1 The DO statement is FORTRAN'S somewhat restricted counterpart of the iteration box. It is introduced in this section and many of its limitations are spelled out to the student. The notion of the range or scope of an iteration or DO-loop is stressed and the CONTINUE statement is explained for marking the end of the scope or range of the DO loop.

You should check the reference manual for the FORTRAN implementation which you will be using and determine if it is one which always causes execution of the scope at least once (most common type) or one which (like the iteration box) checks first to see if the DO-loop variable's initial value has not already exceeded the "final" value. Most IBM processors for FORTRAN II fall in the first category. Nearly all CDC processors for FORTRAN II fall in the second category, etc.

F4-2 The examples of the iteration box used for simple loops in Section 4-2 are mirrored in this section using DO-loops. Much attention must be paid to the problem of getting the student into the habit of having all statements within a loop ultimately lead to the CONTINUE statement of that loop for guaranteeing repetition. The pitfalls are described in connection with Figures F4-5 and F4-6.

The last exercise of Set C brings up a point about the naming of library functions which was not touched on when these functions were first introduced in Table F2-2.

The ABSF is useful for both real and integer arguments. But ABSF

for instance is defined for real arguments only. ABSF(X) for instance returns a real value. Occasionally we may want to express the absolute value of an integer variable or expression. To do this we can use the ABSF function provided the letter X is prefixed to the name, as XABSF.

F4-3 The main purpose of this section, beyond giving the student more practice in converting flow charts having iterations into corresponding FORTRAN programs, is to treat certain details on input-output. These are

1. The use of implied DO-loop notation as list elements of input or output statements. This notation was used in Chapter 3 without explanation there. Here they are explained as a special shorthand related to DO statements.
2. The X-field code for format is introduced and illustrated.

F4-4 Details on the nesting of DO loops are discussed by taking some of the flow charts in Section 4-4 and showing how these may be coded in FORTRAN. The doubly-implied DO-loop notation for input-output list elements is explained in terms of nested DO-loops. This discussion is daggered to suggest its unimportance relative to the other topics that are covered.

Answers to Exercises F4-1

1.

Label	Statement or declaration
	DO 50 I = 1, 50
	READ 60, ID, A, B, C
60	FORMAT(I10,3F10.5)
	D = SQRTF(A**2 + B**2 + C**2)
	PRINT 70, ID, A,B,C,D
70	FORMAT(1H␣,I10,4F10.5)
50	CONTINUE
	STOP
	END

2.

Label	Statement or declaration
100	READ 5, N
5	FORMAT (I5)
	DO 50 I = 1, N
	READ 60, ID, A,B,C
60	FORMAT(I10,3F10.5)
	D = SQRTF(A**2 + B**2 + C**2)
	PRINT 70, ID, A,B,C,D
70	FORMAT(1H␣,I10,4F10.5)
50	CONTINUE
	PRINT 80
80	FORMAT(13H␣END␣OF␣TABLE)
	GO TO 100
	END

3.

Label	Statement or declaration
	LTERM = 2
	NLT = 1
	IS = 1
	DO 30 I = 1, 60
	PRINT 40 I, LTERM, IS
40	FORMAT(1H␣,3I15)
	IS = NLT + IS
	ICOPY = LTERM
	LTERM = LTERM + NLT
	NLT = ICOPY
30	CONTINUE
	STOP
	END

3. Alternate solution with subscripts. (Note the trick of shifting index values to avoid using a zero subscript in FORTRAN.)

Label	Statement or declaration
	DIMENSION IF(63),IS(63)
	IF(2) = 1
	IB = 0
	IS(1) = 0
	DO 60 I = 1,62
	IS(I) = IF(I) + IS(I-1)
	IF(I+1) = IF(I) + IB
	IB = IF(I)
	IF(I - 3) 60, 5, 5
C	TO PRINT FLOW CHART SUBSCRIPT VALUE
	IM1 = I - 1
	PRINT 101, IM1, IF(I), IS(I-2)
101	FORMAT(1H□,3I10)
60	CONTINUE
	STOP
	END

4.

Label	Statement or declaration
C	CARNIVAL WHEEL PROBLEM REPEATED
	DIMENSION NP(4)
	READ 51, (NP(I), I = 1,4)
	READ 51, N
51	FORMAT (4I15)
	ISUM = 0
	DO 20 L = 1, N
	READ 51, NS, M
	K = M+NS - ((M + NS)/4)*4
	ISUM = ISUM + NP(K +)
20	CONTINUE
	PRINT 30, N, ISUM
30	FORMAT(7H AFTER□,I15,
1	30□□SPINS,□YOUR□□NET□□WINNINGS□ARE□,I15,7H□POINTS)
	STOP
	END

5.

Label	Statement or declaration
C	PAYROLL PROBLEM. MAXIMUM NUMBER
C	OF EMPLOYEES ASSUMED TO BE LESS THAN 1000.
	DIMENSION T(1000), R(1000)
	READ 101, N
101	FORMAT(I5)
	PAYROL = 0.
	READ 102, (T(I), I = 1, N)
C	ONE TIME PER CARD. IN HOURS AND HUNDREDTHS
102	FORMAT(F10.2)
	READ 103, (R(I), I = 1, N)
C	UP TO 4 RATES PER CARD. DOLLARS AND CENTS
C	PER HOUR
103	FORMAT (4F10.2)
	DO 50 I = 1, N
	WAGES = R(I)*T(I)
	PAYROL = PAYROL + WAGES
	PRINT 104, I, WAGES
104	FORMAT(1HØ,I10,F15.2)
50	CONTINUE
	PRINT 105, PAYROLL
105	FORMAT(1HØ,F25.2)
	STOP
	END

Answers to Exercises F4-2 Set A

1. Statement 2 should read

```
DO 4 J = 2, N
   ↑
was missing
```

also the statement following the IF should read

```
3 FMAX = ABSF(A(J))
↑
label was missing
```

2. DO statement should read

```
DO 10 J = 4, N,2
      ↑
no comma allowed here!
```

Change MAX, wherever it appears, to a real variable like FMAX.

3. Change FACT to an integer variable, like IFACT or NFACT.
The DO statement is incorrect as written. It should read

```
DO 10 K = 1, N, 1
or  DO 10 K = 1, N
```

4. Either the DO statement should read

```
DO 10 K = 1, N
   ↑
 not L
```

or the PRINT statement should read

```
PRINT 5, L, LTERM
         ↑
       not K
```

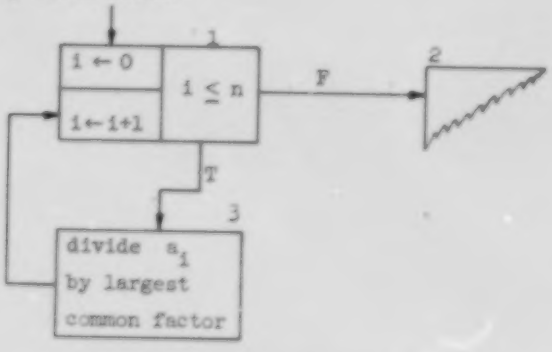
also COPY should be replaced with an integer variable like ICOPY.

Comment on the discussion which immediately follows Exercises F4-2 Set A.

Often we use an iteration box to control the manipulation of a set of elements where the number of elements, represented by N, is previously assigned a value. Sometimes we would like to assign a value to N to suggest the empty set so that when the iteration box is encountered the initial test is false, with the result that no transits through the loop are taken (i.e., no set elements are manipulated).

A good example arises in connection with a problem to be solved in Section 5-4 (Set C Problem 2). There we are to write a procedure called SIMPLIFY, to simplify the integer coefficients of an arbitrary polynomial given its degree n and the coefficients $a_0, a_1, \ldots, a_n$.

Simplification amounts to dividing each coefficient by the absolute value of the largest common factor. However, when the polynomial is identically zero, and this is implied when n has been assigned a negative value like -1, our iteration box allows us to automatically avoid the disaster of dividing by zero. Thus in the loop shown below:



if n = -1 at the time we enter Box 1, we will go immediately to Box 2 without ever entering Box 3, because $0 \leq -1$ is false.

Now, when coding this particular loop in FORTRAN there are two problems.

First, the n + 1 integer coefficients of the polynomial

$$a_0, a_1, a_2, \ldots, a_n$$

cannot be assigned to variables IA(0), IA(1), IA(2), ..., IA(N), because zero subscripts are not allowed in FORTRAN. So we must shift the index of all the coefficients. That is, a_0 is now associated with IA(1), a_1 is associated with IA(2), etc. This means that the initial value for the DO loop index I is 1 instead of zero and the upper limit must have the value

n + 1 instead of n. That's easy to fix. We can write, say

```
      NP1 = N + 1
      DO 50 I = 1,NP1
        :
        :
```

Now we notice the second problem. If N is initially assigned -1, then for the first value of I = 1, I will exceed NP1 which is now 0. If we are using a FORTRAN II processor which generates code that bypasses this test on the first go around, we now need to include the type of test mentioned in the student text. For example:

```
      IF(N) 3, 3, 1
1     NP1 = N + 1
      DO 50 I = 1, NP1
        :
        :
```

Answers to Exercises F4-2 Set B

1.

Label	Statement
	DO 10 I = 1, N
	COPY = P(I)
	P(I) = Q(I)
	Q(I) = COPY
10	CONTINUE

2.

Label	Statement
	DO 10 I = 2, N 2
	COPY = P(I)
	P(I) = Q(I)
	Q(I) = COPY
10	CONTINUE

3.

Label	Statement
	DO 10 I = 5, N, 3
	COPY = P(I)
	P(I) = Q(I)
	Q(I) = COPY
10	CONTINUE

4.

Label	Statement
	ND2 ← N/2
	DO 10 I = 1, ND2
10	Q(I) = P(I)

The alternate solution suggested in the Teachers' Commentary is not possible in FORTRAN. You cannot write DO 10 I = 1, N/2 .

This is illegal.

5.

Label	Statement
	NO2 = N/2
	DO 10 I = 1, NO2
	J = NO2 + I
	Q(I) = P(J)
10	CONTINUE

Note: You cannot write P(NO2 + I) in FORTRAN.

invalid subscript form

6.

Label	Statement
	NO2B = N/2
	IF(N - (N/2) * 2) 3, 6, 3
3	K = NO2B + 1
	GO TO 4
6	K = NO2B
4	DO 10 I = 1, NO2B
	KK = K + I
	Q(I) = P(KK)
10	CONTINUE

7.

Label	Statement
	DO 10 I = 1, K
	LLL = N + 3 - I
	L = N + 1 - I
	P(LLL) = P(L)
10	CONTINUE

FORTRAN doesn't allow us to decrease the value of the loop counter as was done in the equivalent flow chart box.

8(a).

Label	Statement
	SUMCUB = 0.
	DO 10 I = 1, 100
	SUMCUB = SUMCUB + P(I)**3
10	CONTINUE

8(b).

Label	Statement
	SUMNEG = 0.
	DO 10 I = 1, 100
	IF(P(I))4, 10, 10
4	SUMNEG = SUMNEG + P(I)
10	CONTINUE

8(c).

Label	Statement
	SUMCUB = 0.
	SUMNEG = 0.
	SUMBIG = 0.
	DO 10 I = 1, 100
	SUMCUB = SUMCUB + P(I)**3
	IF(P(I))5, 6, 6
5	SUMNEG = SUMNEG + P(I)
6	IF(ABSF(P(I)) - 50.) 10, 10, 7
7	SUMBIG = SUMBIG + ABSF(P(I))
10	CONTINUE

9.

Label	Statement
	COLSUM = 0.
	DO 10 I = 1, 22
	IF(I - 12) 4, 10, 4
4	COLSUM = COLSUM + P(I,K)
10	CONTINUE
	PRINT 15, COLSUM

10.

Label	Statement
	DO 10 J = 1, 27
	P(L,J) = P(L,J) + P(M,J)
10	CONTINUE

11.

Label	Statement
	DO 10 J = 1, 27
	IF(J - K) 3, 10, 3
3	P(L,J) = P(L,J) + 2.0 * P(M,J)
10	CONTINUE

12.

Label	Statement
	DO 10 I = 1, N
	IF(ABSF(P(I)) - 50.) 10, 10, 3
10	CONTINUE
4	IANY = 0
	GO TO 5
3	W = P(I)
	IANY = 1
5	~~~~~~

This is a little more tricky to code in FORTRAN than it may appear at first.

13.

Label	Statement
	W = 50.0
	DO 10 I = 1, N
	J = N - I + 1
	IF(ABSF(P(J)) - 50.0) 10, 10, 4
10	CONTINUE
	GO TO 5
4	W = P(I)

We have had to use the loop counter I as an <u>increasing</u> counter and then use I to define J, which is like a decreasing counter.

14.

Label	Statement
	T = 0.
	DO 10 I = 1, N
	IF(ABSF(P(I)) - ABSF(TM)) 4, 10, 10
4	IF(ABSF(P(I)) - ABSF(T)) 10, 10, 5
5	T = P(I)
10	CONTINUE
	IF(T) 7, 8, 7
7	PRINT 20
20	FORMAT(5H␣NONE)
	STOP
8	~~~~~~~~~~

15. Label | Statement

```
      |  DC 10 I = 1, N
      |  IF (P(I) - TM) 10, 3, 3,
   10 |  CONTINUE
      |  PRINT 20
   20 |  FORMAT(5H␣NONE)
      |  STOP
    3 |  T = P(I)
      |  IK = I + 1
      |  DO 30 K = IK, N
      |  IF(T - P(K)) 6, 30, 30
    6 |  IF(P(K) - TM) 7, 30, 30
    7 |  T = P(K)
   30 |  CONTINUE
    8 |  ~~~~~~~
```

16. Label | Statement

```
      |  SMALL = Q(L, 1)
      |  DO 10 J = 2, N
      |  IF(SMALL - Q(L, J)) 10, 10, 4
    4 |  SMALL = Q(L, J)
   10 |  CONTINUE
```

17. Label | Statement

```
      |  DO 10 K = 1, M
      |  I = M - K + 1
      |  IF(Q(I, IR) - T) 10, 3, 3
   10 |  CONTINUE
      |  IROW = 0
      |  GO TO 5
    3 |  IROW = I
      |  BIG = Q(I, IR)
    5 |  ~~~~~~~~~~~
```

Answers to Exercises F4-2 Set C

1(a)

Label	Statement or declaration
	DIMENSION X(50)
	READ 50, N
50	FORMAT(I3)
	READ 51, (X(I), I = 1, N)
51	FORMAT(5F10.5)
	READ 51, A
	FNUM = X(1) - A
	DO 10 J = 2, N
	FNUM = FNUM * (X(J) - A)
10	CONTINUE
	PRINT 52, FNUM
52	FORMAT(1H□,F15.5)
	STOP
	END

1(b)

Label	Statement or declaration
	DIMENSION X(50)
	READ 50, N
50	FORMAT (I3)
	READ 51, (X(I), I = 1, N)
51	FORMAT (5F10.5)
	READ 52, K, A
52	FORMAT (I3, F10.5)
	FNUM = 1.0
	DO 10 J = 1, N
	IF(J - K) 7, 10, 7
7	FNUM = FNUM * (X(J) - A)
10	CONTINUE
	PRINT 53, FNUM
53	FORMAT(1H□,F15.5)
	STOP
	END

2.

Label	Statement or declaration
	DIMENSION X(50)
	READ 50, N
50	FORMAT(I3)
	READ 51, (X(I), I = 1, N)
51	FORMAT(5F10.5)
	READ 50, K
	DEN = 1.0
	DO 10 J = 1, N
	IF(J - K) 7, 10, 7
7	DEN = DEN * (X(J) - X(K))
10	CONTINUE
	PRINT 52, DEN
52	FORMAT(1H□,F15.5)
	STOP
	END

59

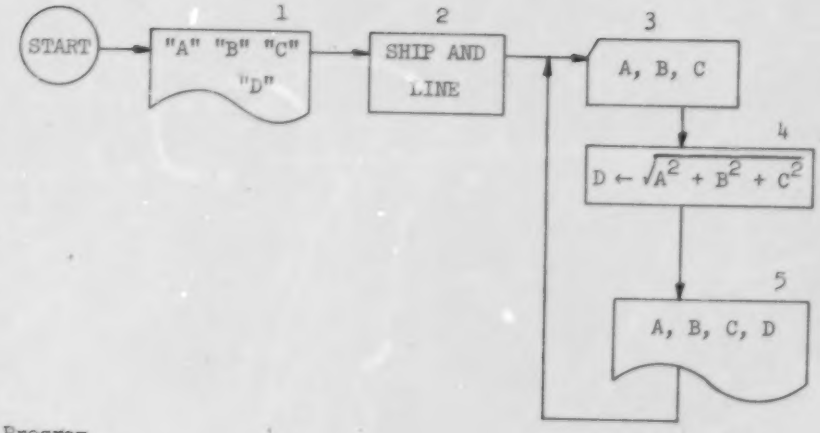
3(b).

Label	Declaration
	DIMENSION NP(4)
	READ 50, (NP(I), I = 1, 4)
50	FORMAT(4I15)
	READ 50, ICV
	ISUM = 0
	DO 10 L = 1,1000
	READ 50, NS, M
	K = M + NS - ((M + NS)/4) * 4
	ISUM = ISUM + NP(K + 1)
	IF(XABSF(ISUM) - ICV) 10, 10, 8
10	CONTINUE
8	PRINT 52, L, ICV, ISUM
52	FORMAT(1H□,3I10)
	STOP
	END

Comment: When we gave the students a list of FORTRAN function names in Table F2-2 we failed to tell them about XABSF.

Answer to Exercise F4-3 Set A

Revised flow chart



Program

Label	Statement or declaration
	PRINT 50
50	FORMAT(1H1, 13X, 1HA, 9X, 1HB, 9X, 1HC, 14X, 1HD)
C	A SEPARATE PRINT STATEMENT IS USED TO PRINT A BLANK
C	LINE
10	PRINT 51
51	FORMAT (1H␣)
3	READ 52, A, B, C
52	FORMAT(3F15.1)
	D = SQRTF(A**2 + B**2 + C**2)
	PRINT 53, A, B, C, D
53	FORMAT(1X, F15.1, 2F10.1, F15.1)
	GO TO 3
	END

Comment: There is another way to create the extra blank line. It is with the use of the slash which is the end-of-record symbol used in format code. We have decided to omit this in our presentation of FORTRAN to the students. Its use does not add enough to the understanding of programming to warrant inclusion--but you should at least be aware of its existence. If we want to skip N lines after printing, we simply add N slashes at the end of the format.

50 FORMAT(1H1,13X,1HA, ~~~ 1HC,14X,1HD/) causes extra line to be skipped

In this way we eliminate the need for Statement 10 and its associated FORMAT declaration.

Answers to Exercises F4-4 Set A

1.

Label	Statement
	BIG = 0
	DO 20 I = 1, M
	DO 10 J = 1, N
	IF(ABSF(BIG) - ABSF(P(I,J))) 5, 10, 10
5	BIG = P(I,J)
10	CONTINUE
20	CONTINUE
	PRINT 50, BIG

2.

Label	Statement
	TLARGE = P(1, 1)
	IROW = 1
	ICOL = 1
	DO 20 I = 1, M
	DO 10 J = 1, N
	IF(TLARGE - P(I,J)) 5, 10, 10
5	TLARGE = P(I,J)
	IROW = I
	ICOL = J
10	CONTINUE
20	CONTINUE
	PRINT 50, TLARGE, IROW, ICOL

3.

Label	Statement
	ZLEAST = 0.
	ZTALY = 0.
	DO 20 I = 1, M, 2
	DO 10 J = 2, N, 2
	IF(P(I,J)) 6, 5, 6
5	ZTALY = ZTALY + 1.
	GO TO 10
6	IF(ZLEAST - P(I,J)) 10, 10, 7
7	ZLEAST = P(I,J)
10	CONTINUE
20	CONTINUE
	PRINT 50, ZLEAST, ZTALY

4.

Label	Statement
	DO 20 I = 2, M
	DO 10 J = 1, N
	P(I,J) = P(I,J) + T * P(1, J)
10	CONTINUE
20	CONTINUE

alternatively (and better)

Label	Statement
	DO 50 J = 1, N
	TEMP = P(1,J)*T
	DO 40 I = 2, M
	P(I,J) = P(I,J) + TEMP
40	CONTINUE
50	CONTINUE

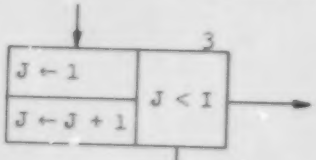
5.

Label	Statement
	DO 20 J = 1, N
	PMIN = P(1, J)
	IROW = 1
	DO 10 I = 2, M
	IF(PMIN - P(I,J)) 10, 5, 5
5	PMIN = P(I,J)
	IROW = I
10	CONTINUE
20	CONTINUE
	PRINT 50, PMIN, J, IROW

6.

Label	Statement
	SUM1 = 0.
	DO 20 I = 2, M
	L = I - 1
	DO 10 J = 1, L
	SUM1 = SUM1 + P(I,J)
10	CONTINUE
20	CONTINUE

Comment 1: Students are liable to be careless on this problem as the iteration box

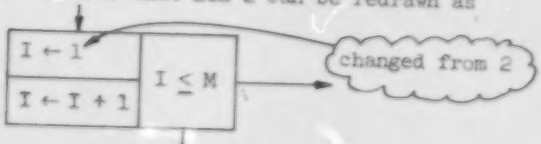


is not the same as DO 20 J = 1, I

and to say DO 20 J = 1, I-1

is illegal. illegal

Comment 2: Refer to the comment in the Teachers Commentary of the Main Text on this problem. It is true that Box 2 can be redrawn as



without changing the effect of the algorithm. However, we cannot make the corresponding change to the corresponding DO statement when using many of the FORTRAN II processors. If we wrote

DO 20 I = 1, M

then when entering the statement

```
DO 10 J = 1, L
```

(where L ← I - 1) there will be no initial test made to see if J which is 1 exceeds L which is 0. If the test were made all would be well.

7.

```
Label   Statement
        SUM 2 = 0.
        MMAX = M - 1
        DO 20 I = 1, MMAX
        IP1 = I + 1
        DO 10 J = IP1, M
        SUM 2 = SUM 2 + P(I,J)
   10   CONTINUE
   20   CONTINUE
```

8.

```
Label   Statement
        IANY = 0
        LIMIT = M - 2
        DO 20 L = 1, LIMIT
        J = M + 1 - L
        ZLAST = P(1,J)
        JM1 = J - 1
        DO 10 I = 2, JM1
        IF(P(I,J) - 2. * ZLAST) 6, 6, 7
    6   ZLAST = P(I, J)
   10   CONTINUE
        GO TO 20
    7   PRINT 50, P(I, J), I, J
        IANY = 1
   20   CONTINUE
        IF(IANY) 11, 10, 11
   10   PRINT 60
   60   FORMAT(5HbNONE)
   11   ~~~~~~~~~~~~~~
```

See the Teachers Commentary of the Main Text for alternate flow chart solutions to Problem 8.

Answers to Exercise F4-4 Set B

1(a)

Label	Statement or declaration
C	NUMBER OF NONCONGRUENT TRIANGLES WHOSE SIDES
C	ARE OF INTEGER LENGTH LESS THAN 100
	IS = 0
	DO 50 I = 1, 100
	JJ = 1 + I/2
	DO 40 J = JJ, I
	IS = IS + 2*J - I
40	CONTINUE
50	CONTINUE
	PRINT 100, IS
100	FORMAT(I10)
	STOP
	END

(b)

Label	Statement or declaration
C	ACCUMULATED PERIMETERS FOR THE S TRIANGLES
C	COUNTED BY PRECEDING PROGRAM
	IP = 0
	DO 50 I = 1, 100
	JJ = 1 + I/2
	DO 40 J = JJ, I
	KK = I - J + 1
	DO 30 K = KK, J
	IP = IP + I + J + K
30	CONTINUE
40	CONTINUE
50	CONTINUE
	PRINT 100, IP
100	FORMAT(I10)
	STOP
	END

Answers to Exercises F4-4 Set C

1.

Label	Statement
	READ 50, N
50	FORMAT(I5)
	FN = N
	LIMIT = SQRTF(FN)
	DO 10 K = 2, LIMIT
3	IF(N - (N/K) * K) 10, 4, 10
4	PRINT 51, K
51	FORMAT(1H⌴,I5)
	N = N/K
	GO TO 3
10	CONTINUE
	IF(N - 1) 7, 8, 7
7	PRINT 51, N
8	STOP
	END

2.

Label	Statement or Declaration	
	DIMENSION A(500)	
	READ 50, N	
50	FORMAT(I5)	
	READ 51, (A(I), I = 1, N)	
51	FORMAT(4F10.5)	
	NM1 = N - 1	
	DO 20 J = 1, NM1	
	IF(A(J) - A(J + 1)) 20, 20, 5	
5	COPY = A(J)	
	A(J) = A(J + 1)	
	A(J + 1) = COPY	
	LIMIT = J - 1	
	IF(LIMIT) 30, 30, 40	→ prevents inner DO from being executed when LIMIT = 0
30	GO TO 20	
40	DO 10 L = 1, LIMIT	
	K = LIMIT + 1 - L	
	IF(A(K) - A(K + 1)) 20, 20, 8	
8	COPY = A(K)	
	A(K) = A(K + 1)	
	A(K + 1) = COPY	
10	CONTINUE	
20	CONTINUE	
	PRINT 52, (A(I), I = 1, N)	
52	FORMAT(1H⌴,4F10.5)	
	STOP	
	END	

3.

Label	Statement
	DIMENSION A(500)
	READ 50, N
50	FORMAT(I5)
	READ 51, (A(K), K = 1, N)
51	FORMAT (4F10.5)
	NM1 = N - 1
	DO 20 I = 1, NM1
	NMI = N - I
	DO 10 J = 1, NMI
	IF(A(J) - A(J + 1)) 10, 10, 5
5	COPY = A(J)
	A(J) = A(J + 1)
	A(J + 1) = COPY
10	CONTINUE
20	CONTINUE
	PRINT 52, (A(K), K = 1, N)
52	FORMAT(4F10.5)
	STOP
	END

4.

Label	Statement or declaration
	DIMENSION A(100), IB(100)
	READ 50, N
50	FORMAT (I5)
	READ 51, (A(I), I = 1, N)
51	FORMAT (4F10.5)
	MAXINC = 1
	DO 20 J = 1, N
	B(J) = 1
	JM1 = J - 1
	IF(JM1) 30, 30, 40
30	GO TO 13
40	DO 10 K = 1, JM1
	IF(A(K) - A(J)) 10, 10, 5
5	IF(IB(J) - (IB(K) + 1)) 6, 10, 10
6	IB(J) = IB(K) + 1
10	CONTINUE
13	IF(MAXINC - IB(J)) 14, 20, 20
14	MAXINC = B(J)
20	CONTINUE
	PRINT 52, MAXINC
52	FORMAT(1H□,I5)
	STOP
	END

→ prevents inner DO from being executed when JM1 = 0

Chapter TF5

SUBPROGRAMS

Use of the computed GO TO statement

FORTRAN II compilers for the smaller computers sometimes do not allow us to define FUNCTION subprograms. All FORTRAN compilers do, however, provide the so-called computed GO TO statement. With this type of statement we can simulate the action of a functional reference as part of the main program.

We give first a brief definition of the computed GO TO and then use it in simulating the FUNCTION subprograms given in Figure F5-2.

Suppose the integer variable K is assigned a value, k, in the range $1 \leq k \leq m$ where m is some small number like 5. Now at certain points in a program suppose we would like to GO TO one of a set of m statements whose labels are $s_1, s_2, \ldots, s_m$. With each value k of K we associate the statement label s_k so that if k is the value of K, we GO TO label s_k. The FORTRAN statement that accomplishes this has the form

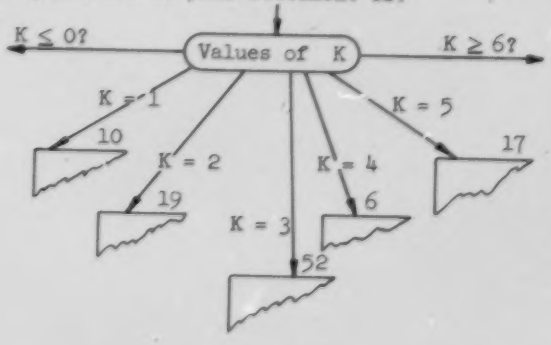
GO TO $(s_1, s_2, s_3, \ldots, s_m)$, K

Example.

GO TO (10, 19, 52, 6, 17), K

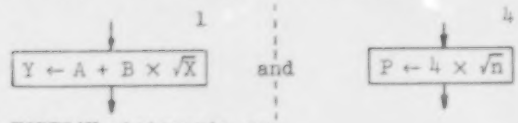
Means: Transfer control to the statement labeled either 10, 19, 52, 6, or 17 when the value of K is either 1, 2, 3, 4, or 5 respectively. If the value of K is not in the proper range, no guarantee is given as to what will be the next task executed.

The flow chart equivalent of this statement is:



The computed GO TO provides us with a convenient switching mechanism for returning to a point of call, simulating the automatic return that is provided by the RETURN statement in a subprogram.

We now illustrate the switching mechanism for a program much like the main flow chart shown in Figure 5-6 but which contains imbedded in it the heart of the algorithm used in the reference flow chart in the same figure. There are two points of call at box 1 and at box 4. Suppose the detail of these boxes were



The corresponding FORTRAN statements are

```
 1 | Y = A + B * SQRTF(X)

 4 | TEMP = N
   | P = 4. * SQRTF(TEMP)
```

Suppose the square root algorithm is placed in the program beginning at the statement labeled 100. Then we might revise statements 1 and 4 as

In place of Statement 1

```
 1 | I = 1
   | ARG = X
   | GO TO 100
51 | Y = A + B * ROOT
   :
```

In place of Statement 4

```
 4 | I = 2
   | ARG = N
   | GO TO 100
61 | P = 4. * ROOT
```

The coding at statement 100 would end in a computed GO TO as shown below:

```
   :
100 | G = 1.0
  2 | H = 0.5 * (G + ARG/G)
    | IF(ABSF(H - G) - .0001) 5, 10, 10
 10 | G = H
    | GO TO 2
  5 | ROOT = H
    | GO TO (51, 61), I
```

Returns control to Statement 51 or 61 depending on the point of call.

Answers to Exercises F5-1

1.

```
  | FUNCTION CUBRT(A)
  | G = 1
2 | H = (2. * G + A/G**2)/3.
  | IF(ABSF(H - G) - .0001) 5, 4, 4
4 | G = H
  | GO TO 2
5 | CUBRT = H
  | RETURN
  | END
```

alternate

```
  | FUNCTION CUBRT(A)
  | G = 1
2 | CUBRT=(2.*G+A/G*G)/3.
  | IF(ABSF(CUBRT-G)-.0001)
  |    5, 4, 4
4 | G = CUBRT
  | GO TO 2
5 | RETURN
  | END
```

2.

```
      FUNCTION FUNCT(X)
      FUNCT = (3. * X - 2.) * X + 1.
      RETURN
      END
```

3.

```
      FUNCTION ABSOL(X)
      IF (X) 2, 3, 3
    2 ABSOL = -X
      GO TO 4
    3 ABSOL = X
    4 RETURN
      END
```

Answers to Exercises F5-3 Set A

1(a).

```
      FUNCTION FUNCT(X,Y)
      FUNCT = ((X**3 + Y)**2 + 5.)/(ABSF(X) + 2.)
      RETURN
      END
```

(b).

```
      Y = FUNCT(R,S) + 6. * T
```

2.

```
      FUNCTION RIGHT(A, B, C)
      IF (C) 7, 7, 2
    2 IF (B) 7, 7, 3
    3 IF (A) 7, 7, 4
    4 IF(C * C - (A * A + B * B)) 5, 8, 5
    5 IF(A * A - (B * B + C * C)) 6, 8, 6
    6 IF(B * B - (A * A + C * C)) 7, 8, 7
    7 RIGHT = 0
      RETURN
    8 RIGHT = 1
      RETURN
      END
```

3(a).

```
      FUNCTION XMAX (X, Y, Z)
      XMAX = X
      IF(XMAX - Y) 5, 3, 3
    3 IF(XMAX - Z) 6, 4, 4
    4 RETURN
    5 XMAX = Y
      GO TO 3
    6 XMAX = Z
      RETURN
      END
```

(b).

```
      READ 11, A, B, C
      XLRGST = XMAX (A, B, C)
      PRINT 11, XLRGST
      STOP
   11 FORMAT (3F15.8)
      END
C     DEFINITION OF FUNCTION XMAX GOES HERE
      END
```

4. Let a zero value for IQUAD represent an error exit for a point on a coordinate axis.

```
      |FUNCTION IQUAD(X, Y)
      |IF(X) 5, 8, 2
    2 |IF(Y) 4, 8, 3
    3 |IQUAD = 1
      |RETURN
    4 |IQUAD = 4
      |RETURN
    5 |IF(Y) 7, 8, 6
    6 |IQUAD = 2
      |RETURN
    7 |IQUAD = 3
      |RETURN
    8 |IQUAD = 0
      |RETURN
      |END
```

5.

```
      |FUNCTION INTSCT(X1,Y1,R1,X2,Y2,R2)
      |IF(R1) 12, 12, 2
    2 |IF(R2) 12, 12, 3
    3 |DIST = SQRTF((X2-X1)**2 + (Y2-Y1)**2)
      |IF(DIST) 5, 10, 5
    5 |IF(DIST - (R1+R2)) 51, 7, 51
   51 |IF(DIST - ARSF(R1-R2)) 6, 7, 6
    6 |IF(DIST - ABSF(R1-R2)) 8, 61, 61
   61 |IF(R1 + R2 - DIST) 8, 9, 9
    9 |INTSCT = 2
      |RETURN
   10 |IF(R1-R2) 8, 11, 8
    8 |INTSCT = 0
      |RETURN
   11 |INTSCT = 100
      |RETURN
    7 |INTSCT = 1
      |RETURN
   12 |INTSCT = -1
      |RETURN
      |END
```

6.

```
      |FUNCTION ANYRT(Y,N)
      |XN = N
      |G = 1
      |DO 10 I = 1, 10
      |     H = Y/G**(N - 1)
      |     IF(ABSF(G - H) - .0001) 6, 5, 5
    5 |     G = ((XN - 1.) * G + H)/XN
   10 |     CONTINUE
    6 |ANYRT = G
      |RETURN
      |END
```

7(a).

```
      FUNCTION TIRATE(N,R,TL)
      TIRATE = TL
      RATE = .01 * R
      DO 10 I = 1, N
         TIRATE = TIRATE * (1. + RATE)
   10    CONTINUE
      RETURN
      END
```

Alternate solution:

```
      FUNCTION TIRATE (N,R,TL)
      TIRATE = TL * (1. + R/100.) ** N
      RETURN
      END
```

Comment:

There is in FORTRAN II a so-called "statement function" form of function definition which would be applicable to the alternate solution. (We do not feel it is important enough to inject this in the student text.) Using this statement function form, one can write the whole function definition as a single declaration:

```
C     THE FOLLOWING IS THE COMPLETE FUNCTION DEFINITION
      TIRATE (N, R, TL) = TL * (1. + R/100.) ** N
```

7(b).

```
      ISWTCH = 0
      X = 100.
      Y = 100.
      DO 100 I = 1, 36
           X = X * 1.01
           Y = Y + 1.125
           IF (X-Y) 6, 6, 5
    5      ISWTCH = 1
           MONTH = I
    6      IF ((I/12) * 12 - I) 100, 7, 100
    7      PRINT 101, I, X, Y
  100      CONTINUE
      IF ((ISWTCH) - 1) 10, 9, 10
    9 PRINT 102, MONTH
   10 STOP
  101 FORMAT(1H□,□I5,2F10.2)
  102 FORMAT(10H□MONTHS□ □,I2)
      END
```

7(b). Alternate solution

```
      |ISWTCH = 0
      |X = 100.
      |Y = 100.
      |DO 100 I = 1,36
      |     X = TIRATE (I, 1., 100.)
      |
      |     Y = Y + 1.125
      |
      |     IF(X - Y) 6, 6, 5
    5 |     ISWTCH = 1
      |     MONTH = I
    6 |     IF((I/12) * 12 - I) 100, 7, 100
    7 |     PRINT 101, I, X, Y
  100 |     CONTINUE
      |IF(ISWTCH - 1) 10,9, 10
    9 |PRINT 102, MONTH
   10 |STOP
  101 |FORMAT(1H□,I5,2F10.2)
  102 |FORMAT(10H□MONTHS□=□,I2)
      |END
```

7(c). Alternate solution

```
      |X = 100.
      |NX = 1
    3 |X = X * 1.01
      |IF(X - 200.) 6, 5, 5
    5 |PRINT 101, NX, X
      |STOP
    6 |NX = NX + 1
      |GO TO 3
  101 |FORMAT(5H□NX□=,I5,3HX□=,F10.2)
      |END
```

Answers to Exercises F5-3 Set B

1.

Label		Statement or declaration
C		GREATEST COMMON DIVISOR AS AN INTEGER-VALUED
C		FUNCTION CALLED KGCD
		FUNCTION KGCD(KA, KB)
		IF(KA - KB) 5, 5, 4
	4	KR = KB
		KB = KA
		KA = KR
	5	IF(KA) 6, 7, 6
	6	KR = KB - (KB/KA) * KA
		KB = KA
		KA = KR
		GO TO 5
	7	KGCD = KB
		RETURN
		END

2.

Label	Statement or declaration
C	GREATEST COMMON FACTOR ALGORITHM
C	CALLED KGCF. IT CALLS ON KGCD
	FUNCTION KGCF(KA,KB,KC)
	KX = KGCD(KA,KB)
	KGCD = KGCD(KX,KC)
	RETURN
	END

3(a).

Label	Statement or declaration
C	NUMBER OF NON-SIMILAR TRIANGLES AMONG
C	THOSE OF INTEGER LENGTH LESS THAN 100
	IS = 0
	DO 50 I = 1, 100
	JJ = 1 + I/2
	DO 40 J = JJ, I
	KK = I - J + 1
	DO 30 K = KK, J
	IF(KGCF(I,J,K) - 1) 30, 6, 30
6	IS = IS + 1
30	CONTINUE
40	CONTINUE
50	CONTINUE
	PRINT 100, IS
100	FORMAT(I10)
	STOP
	END

3(b). Statement 6 to be replaced by

6	NS = NS + I + J + K

Comment: In case students run this problem and also Problem 4 on the computer, you will probably want them to cut down the size of the problem, otherwise excessive computer time may be required. For example, triangles whose lengths are less than 50 might be illustrative enough. Likewise in Problem 4, you could consider all numbers less than 10^6 instead of 10^9.

4.

Label		Statement or declaration
		DIMENSION ICUBE (999)
		DO 60 I = 1, 999
		ICUBE(I) = I ** 3
		DO 50 J = 1, I
		ITEST = ICUBE(I) + ICUBE(J)
		IF(ITEST - 1000000000) 6, 60, 60
6		K = I - 1
		L = J + 1
7		IF(L - K) 8, 8, 50
8		IF(ICUBE(L) + ICUBE(K) - ITEST) 10, 9, 11
9		IF(KGCF(J, K, L) - 1) 13, 12, 13
10		L = L + 1
		GO TO 7
11		K = K - 1
		GO TO 7
12		PRINT 101, I, J, ITEST, K, L
101		FORMAT(I6,9H□CUBED□+□,I6,9H□CUBED□=□,
	1	I10,3H□=□,I6,9H□CUBED□+□,I6,6H□CUBED)
13		L = L + 1
		K = K - 1
		GO TO 7
50		CONTINUE
60		CONTINUE
		STOP
		END

Answers to Exercises P5-4 Set A

1.

```
      SUBROUTINE ABSOL (X,ABSX)
      IF(X) 2, 3, 3
    2 ABSX = -X
      RETURN
    3 ABSX = X
      RETURN
      END
```

2(a).

```
      SUBROUTINE CXADD (A1, B1, A2, B2, A, B)
      A = A1 + A2
      B = B1 + B2
      RETURN
      END
```

(b).

```
      SUBROUTINE CXSUBT (A1, B1, A2, B2, A, B)
      A = A1 - A2
      B = B1 - B2
      RETURN
      END
```

(c).

```
      SUBROUTINE CXMULT (A1, B1, A2, B2, A, B)
      A = A1 * A2 - B1 * B2
      B = A1 * B2 + A2 * B1
      RETURN
      END
```

(d).

```
      SUBROUTINE CXDIV (A1, B1, A2, B2, A, B)
      DENOM = A2 * A2 + B2 * B2
      A = (A1 * A2 + B1 * B2)/DENOM
      B = (A2 * B1 - A1 * B2)/DENOM
      RETURN
      END
```

(e).

```
      READ 11, A1, B1, A2, B2, OPER
      PRINT 11, A1, B1, A2, B2, OPER
   11 FORMAT (5F10.4)
      IF (OPER - 2.) 4, 5, 6
    4 CALL CXADD (A1, B1, A2, B2, A, B)
      GO TO 9
    5 CALL CXSUBT (A1, B1, A2, B2, A, B)
      GO TO 9
    6 IF(OPER - 3.) 7, 7, 8
    7 CALL CXMULT (A1, B1, A2, B2, A, B)
      GO TO 9
    8 CALL CXDIV (A1, B1, A2, B2, A, B)
    9 PRINT 12, A, B
   12 FORMAT (1H□F10.4,1H + F10.4,1HI)
      GO TO 1
      END
C     DEFINITION OF CXADD, CXSUBT, CXMULT, CXDIV GO HERE
      END
```

3. A normal exit is indicated by ERROR = 0. If ERROR = 1, then $K \leq 0$.

```
      SUBROUTINE SORT 2(K, A, B, ERROR)
      DIMENSION A(1000), B(1000)
      IF(K) 6, 6, 2
    2 ERROR = 0
      LAST = K - 1
    3 DO 50 I = 1, LAST
      IF(A(I) - A(I+1)) 50, 50, 5
   50 CONTINUE
      RETURN
    5 COPY = A(I)
      A(I) = A(I + 1)
      A(I + 1) = COPY
      COPY = B(I)
      B(I) = B(I + 1)
      B(I + 1) = COPY
      GO TO 3
    6 ERROR = 1
      RETURN
      END
```

4(a).

```
      SUBROUTINE COUNT (N,NOFAC)
      IF(N) 5, 5, 1
    1 NOFAC = 0
      XN = N
      IBOUND = SQRTF(XN) - 1
      DO 3 K = 1, IBOUND
      IF(N - K * (N/K)) 3, 4, 3
    4 NOFAC = NOFAC + 2
    3 CONTINUE
      IF (N - K * K) 8, 7, 8
    7 NOFAC = NOFAC + 1
    8 RETURN
    5 NOFAC = -1
      RETURN
      END
```

Note: Since the argument of SQRTF must be floating point, we use XN as the floating point representation of N.

(b).

```
      DO 2 N = 1,1000
      CALL COUNT (N, IFAC)
      IF(IFAC - 2) 2, 3, 2
    3 PRINT 11, N
   11 FORMAT (I10)
    2 CONTINUE
      STOP
      END
C     DEFINITION OF SUBROUTINE COUNT GOES HERE
      END
```

5(a).

```
      SUBROUTINE ALQUOT (NUMBER, N, IPART)
      DIMENSION IPART (50)
      IF (NUMBER) 8, 8, 2
    2 N = 1
      IPART (1) = 1
      IF (NUMBER - 3) 9, 9, 4
    4 XN = NUMBER
      IBOUND = SQRTF(XN)
      DO 10 K = 2, IBOUND
      IF (NUMBER - K * (NUMBER/K)) 10, 7, 10
    7 N = N + 2
      IPART(N - 1) = K
      IPART (N) = NUMBER/K
   10 CONTINUE
      IF(PART(N) - PART(N - 1)) 9, 11, 9
   11 N = N - 1
      RETURN
    8 N = -1
    9 RETURN
      END
```

(b).

```
      DIMENSION IA(50)
      DO 10 I = 1, 500
      CALL ALQUOT (I, N, IA)
      IF(N) 9, 9, 4
    4 ISUM = 0
      DO 6 J = 1, N
      ISUM = ISUM + IA(J)
    6 CONTINUE
      IF (I - ISUM) 10, 8, 10
    8 PRINT 21, I
   10 CONTINUE
      GO TO 11
    9 PRINT 22
   11 STOP
   21 FORMAT (I10)
   22 FORMAT (11H IMPOSSIBLE)
      END
C     DEFINITION OF SUBROUTINE ALQUOT GOES HERE
      END
```

The first five perfect numbers are 6, 28, 496, 8128, 33550336.

TP5

5(c). Let B be an array. B_i is the sum of the aliquot parts for number i.

```
      |  DIMENSION IA(50), IB(500)
      |  DO 50 I = 1, 500
      |  CALL ALQUOT (I,N,IA)
      |  IF(N) 11, 11, 4
    4 |  ISUM = 0
      |  DO 6 J = 1, N
      |  ISUM = ISUM + IA(J)
    6 |  CONTINUE
      |  IB(I) = ISUM
      |  IF(ISUM - I) 9, 50, 50
    9 |  IF(IB(ISUM) - I) 50, 10, 50
   10 |  PRINT 21, ISUM, I
   50 |  CONTINUE
      |  GO TO 12
   11 |  PRINT 22
   12 |  STOP
   21 |  FORMAT (2I10)
   22 |  FORMAT (11H IMPOSSIBLE)
      |  END
C     |  DEFINITION OF SUBROUTINE ALQUOT GOES HERE
      |  END
```

Alas, the only pair of friendly numbers less than 500 are 220 and 284.

Answers to Exercises F5-4 Set B

1.
```
C     THE LEAST FUNCTION IS CALLED TLEAST
      FUNCTION TLEAST (N,A)
      DIMENSION A(100)
      S = A(1)
      DO 60 I = 1, N
           IF(A(I) - S) 60, 60, 4
    4     S = A(I)
   60     CONTINUE
      TLEAST = S
      RETURN
      END
```

2.
```
C     THE SUBLEAST FUNCTION IS CALLED MLEAST
      FUNCTION MLEAST (N, A)
      DIMENSION A(100)
      S = A(1)
      K = 1
      DO 60 I = 1, N
           IF(A(I) - S) 60, 60, 4
    4     S = A(I)
          K = I
   60     CONTINUE
      MLEAST = K
      RETURN
      END
```

3.
```
C     THE PROCEDURE CALLED MARKS
      SUBROUTINE MARKS (N,A,S,K)
      DIMENSION A(100)
      S = A(1)
      K = 1
      DO 60 I = 1, N
           IF(A(I) - S) 60, 60, 4
    4     S = A(I)
          K = I
   60     CONTINUE
      RETURN
      END
```

Answers to Exercises F5-4 Set C

1.

Label	Statement or declaration
C	THE PROCEDURE CALLED DEGREE.
C	FOR FORTRAN'S SAKE, SUBSCRIPTS ARE SHIFTED BY ONE.
	SUBROUTINE DEGREE (N, IA)
	DIMENSION IA(100)
1	IF(IA(N+1)) 4, 2, 4
2	N = N - 1
	IF(N) 4, 1, 1
4	RETURN
	END

TP5

2.

Label	Statement or declaration
C	PROCEDURE CALLED SMPLFY TO
C	SIMPLIFY THE COEFFICIENTS OF AN NTH
C	DEGREE POLYNOMIAL, N LESS THAN 100.
C	(FOR FORTRAN'S SAKE, ALL SUBSCRIPTS ARE
	SHIFTED BY ONE.)
	SUBROUTINE SMPLFY (N,IA)
	DIMENSION IA(100)
C	TEST TO SEE IF DEGREE IS NEGATIVE. IF SO, DO NOTHING
	IF(N - (-1)) 7, 7, 1
1	ID = ABSF(IA(1))
	NPLUS1 = N + 1
	DO 70 I = 2, NPLUS1
	ID = KGCD(ID, ABSF(IA(I)))
	IF(ID - 1) 70, 7, 70
70	CONTINUE
	DO 60, I = 1, NPLUS1
	IA(I) = IA(I)/ID
60	CONTINUE
7	RETURN
	END

*3.

Label	Statement or declaration
C	PROCEDURE RDCMOD FOR REDUCING
C	AN NTH DEGREE POLYNOMIAL, A(X),
C	MODULO AN MTH DEGREE POLYNOMIAL B(X)
C	WHERE M LESS THAN OR EQUAL TO N,
C	AND N LESS THAN 100.
C	(FOR FORTRAN'S SAKE, ALL SUBSCRIPTS ARE
C	SHIFTED BY ONE FROM THE VALUES IN THE
	FLOW CHART)
	SUBROUTINE RDCMOD (N,M,IA,IB)
	DIMENSION IA(100), IB(100)
	IF(M) 10, 10, 2
11	IF(N-M) 10, 11, 11
2	NP1 = N + 1
	MP1 = M + 1
	IX = KGCD(IA(NP1), IB(MP1))
	IC = IB(MP1)/IX
	ID = IA(NP1)/IX
	DO 60 I = 2, NP1
C	FORM VALID SUBSCRIPT FOR N - I
	NMINI = N - I
	IF(I - MP1) 5, 5, 6
C	FORM VALID SUBSCRIPT FOR M - I
5	MMINI = M - I
	IA(NMINI) = IC * IA(NMINI) - ID*IB(MMINI)
	GO TO 60
6	IA(NMINI) = IC * IA(NMINI)
60	CONTINUE
	N = N - 1
	CALL DEGREE (N, IA)
	CALL SMPLFY (N, IA)
	GO TO 11
10	RETURN
	END

*4.

Label	Statement or declaration
C	PROGRAM FOR FINDING THE GREATEST COMMON
C	DIVISOR OF TWO POLYNOMIALS
C	THE TWO POLYNOMIALS IA AND
C	IB HAVING DEGREES N AND
C	M, RESPECTIVELY, AND EACH IS
C	LESS THAN 100.
C	FOR FORTRAN'S SAKE, INDEXES ARE SHIFTED BY ONE
	DIMENSION IA(100), IB(100)
	READ 101, N, M
	MP1 = N + 1
	MP1 = M + 1
	READ 101, (IA(I), I = 1, MP1)
	READ 101, (IB(I), I = 1, MP1)
	CALL DEGREE(N, IA)
	CALL SMPLFY(N, IA)
	CALL DEGREE(M, IB)
	CALL SMPLFY(M, IB)
	ISW = 0
7	CALL RDCMOD(N,M,IA,IB)
	IT = N
11	IF(IT) 15, 14, 12
12	ISW = 1 - ISW
	IF(ISW - 1) 7, 8, 7
8	CALL TDCMOD (M, N, IB, IA)
	IT = M
	GO TO 11
14	PRINT 102
	STOP
15	IF(ISW) 17, 16, 17
16	PRINT 103, (B(I), I = 1, MP1)
	STOP
17	PRINT 103, (A(I), I = 1, MP1)
	STOP
101	FORMAT(10I6)
102	FORMAT(2H 1)
103	FORMAT(1H , 10I6)
	END

Supplementary Exercises in the Teacher's Commentary at the end of Section 5-4.

Here are the FORTRAN programs for the flow charts given in the solution set to these exercises.

1.

```
      SUBROUTINE INTO (N, NA, NB, NDEC)
      DIMENSION NA(100)
      NDEC = 0
      DO 3 I = 1, N
      NDEC = NDEC + NA(I) * NB**(N - I)
    3 CONTINUE
      RETURN
      END
```

2(a). In many implementations of FORTRAN there is no parallel to the assignment $COMP_1 \leftarrow$ "0". Instead we must input the alphanumeric elements of vector NCOMP in the main program and include NCOMP in the parameter list for function IDENT.

```
      SUBROUTINE IDENT (K, NA, NCOMP, IVALUE)
      DIMENSION NA(100), NCOMP(16)
      DO 3 J = 1, 16
      IF(NA(K) - NCOMP(J)) 3, 4, 3
    3 CONTINUE
      PRINT 11
   11 FORMAT (20H␣INCORRECT␣CHARACTER)
      STOP
    4 IVALUE = J - 1
      RETURN
      END
```

3.

```
      SUBROUTINE NHEXDC (N, NA, NCOMP, NDEC)
      DIMENSION NA(100), NCOMP (16)
      NDEC = 0
      DO 4 I = 1, N
      CALL IDENT (I, NA, NCOMP, NVALUE)
    4 NDEC = NDEC + NVALUE * 16**(N - I)
      RETURN
      END
```

4.

```
      SUBROUTINE OUT (ND, NB, IR, M)
      DIMENSION IR(100)
      M = 1
    2 NQ = ND/NB
      IR(M) = ND - NQ * NB
      IF(NQ) 5, 5, 4
    4 M = M + 1
      ND = NQ
      GO TO 2
    5 RETURN
      END
```

5.

```
      DIMENSION NA(100), NR (100)
    1 READ 11, NB1, NB2, N, (NA(K), K = 1, N)
      CALL INTO (N, NA, NB1, NBAS10)
      CALL OUT (NBAS10, NB2, NR, M)
C     REVERSE ORDER OF NR ENTRIES IN NEXT SIX STEPS
      MIDDLE = (M + 1)/2
      DO 10 I = 1, MIDDLE
      K = M + 1 - I
      NEED = NR(I)
      NR(I) = NR(K)
      NR(K) = NEED
   10 CONTINUE
      PRINT 12, (NR(I), I = 1, M)
      GO TO 1
   11 FORMAT (3I3, (71I1))
   12 FORMAT (1H␣100I1)
      END
C     DEFINITION OF INTO AND OUT GO HERE
      END
```

6(a). As in the hexadecimal-to-decimal-conversion exercise (Problem 2), the alphanumeric comparison vector, in this case ROMAN, must be input into the main program and then transferred to the subroutine subprogram as one of the components of the parameter list. Input the elements of ROMAN in this order: I, V, X, L, C, D, M.

```
      SUBROUTINE RNUM (N, A, ROMAN, NUM)
      DIMENSION A(20), ROMAN(7), NVALUE(7)
      NVALUE(1) = 1
      NVALUE(2) = 5
      NVALUE(3) = 10
      NVALUE(4) = 50
      NVALUE(5) = 100
      NVALUE(6) = 500
      NVALUE(7) = 1000
      NUM = 0
      LAST = 8
    4 DO 50 K = 1, N
      DO 6 I = 1, 7
      IF(A(K) - ROMAN(I))6, 7, 6
    6 CONTINUE
      PRINT 21
      STOP
    7 IF(LAST - I) 8, 9, 9
    8 NUM = NUM - 2 * NVALUE(LAST) + NVALUE(I)
      GO TO 10
    9 NUM = NUM + NVALUE(I)
   10 LAST = I
   50 CONTINUE
      RETURN
   21 FORMAT (20H□INCORRECT□CHARACTER)
      END
```

(b).

```
      DIMENSION A(20), B(20), ROMAN(7)
      READ 25, (ROMAN(I), I = 1, 7)
    1 READ 21, N, (A(K), K = 1, N)
      PRINT 21, N, (A(K), K = 1, N)
      READ 21, M, (B(K), K = 1, M)
      PRINT 21, M, (B(K), K = 1, M)
      CALL RNUM (N, A, NUM1)
      CALL RNUM (M, B, NUM2)
      SUM = NUM1 + NUM2
      PRINT 22, SUM
      GO TO 1
   21 FORMAT (I10, 20A1)
   22 FORMAT (7H□SUM□ =□F10.0)
   25 FORMAT (7A1)
      END
C     DEFINITION OF SUBROUTINE RNUM GOES HERE
      END
```

Supplementary remarks on assignment of values to non-local variables

The following should help you to visualize how the information on non-local variables is transmitted to a FORTRAN subprogram. Suppose a calling argument is a single variable, like T, and suppose it is matched with a dummy variable, say X in the subprogram. At the time the subprogram is executed, the subprogram "takes note of" the address where the value of T may be found (i.e., location of the window box labeled T). If the calling argument is an expression like 2. * T, the subprogram will instead take note of the address where the value of 2. * T has been temporarily stored. The programmer is normally not aware that the compiler, in generating the code for a call on a subprogram, generates instructions to evaluate argument expressions and to save their values in "temporary" storage locations. Although the programmer is not told where these "temporaries" are located, their addresses are nevertheless made available to the subprogram each time the subprogram is called.

We can now begin to perceive how the foregoing explanation might relate to our sealed chamber models with slips of paper or boxes being dropped into the funnel. If a subprogram is told where the value of an argument is located, it can alter the value of the argument. Hence this corresponds in our model to dropping into the funnel the window box that contains the value of the argument. If the argument is 2. * T, however, the subprogram is not told where T is located, only where the value of computation 2. * T is located. Hence the subprogram cannot alter the value of T. In a way this suggests the protection which is inherent in the slip of paper concept. Strictly speaking we could "protect" any variable used as a calling argument to a FORTRAN function by first assigning the value of that variable to an auxiliary variable and then using that auxiliary variable as the argument. In this way no matter what the function did with the window box it received, it could not "harm" the value of the original variable.

Example: Suppose the FUNCTION is called FUN and it has arguments A, B, C. Then one way to simulate the slip of paper concept of protection is by writing

```
T1 = A
T2 = B
T3 = C
 Z = FUN(T1, T2, T3)
```

instead of

```
Z = FUN(A, B, C)
```

A more sensible way to accomplish the same objective (and actually a safer and more efficient way) is to let the subprogram itself reassign incoming values of the dummy variables to new auxiliary variables that are strictly local to the subprogram. After this reassignment the auxiliary variables are used in place of the dummy arguments. Thus for the above example, the call to FUN could be safely written as:

```
Z = FUN (A, B, C)
```

provided the definition of FUN is coded something like

```
FUNCTION FUN (DUMMY1, DUMMY2, DUMMY3)
ARG1 = DUMMY1
ARG2 = DUMMY2
ARG3 = DUMMY3
  :
END
```

Function name arguments in FORTRAN II processors

Some of the processors for the larger computers like the IBM 7040 do in fact provide an "improvised" way to define functions having function name arguments (but not label arguments). To use a function name as an argument in such processors, it is necessary to provide a special card in the calling program which lists the function names that are to be used as arguments. This card has the letter F punched in Column 1.

There are other details regarding the dropping of the terminal letter F on library subroutines that are used as arguments. Such details are messy. Even if you have this capability available to you in the laboratory one could question the wisdom of teaching it to students at this level. In more advanced languages function name arguments are handled in a more natural way. For more details you should, of course, consult the appropriate reference manual.

Answers to Exercises P5-5

1.
```
C     FORTRAN SUBROUTINE FOR TWO SIMULTANEOUS EQUATIONS
      SUBROUTINE ROOTS2(A1,B1,C1,A2,B2,C2,X1,X2,N)
      DENOM = A1 * B2 - A2 * B1
      IF(DENOM) 3, 5, 3
    3 X1 = (B1 * C2 - B2 * C1)/DENOM
      X2 = (A1 * C2 - A2 * C1)/DENOM
      N = 0
      RETURN
    5 N = 1
      RETURN
      END
```

2(a).

```
C     A SUBROUTINE SUBPROGRAM FOR REAL
C     ROOTS OF QUADRATIC EQUATIONS
      SUBROUTINE ROOTS (A, B, C, N, X1, X2)
      IF(A) 2, 11, 2
    2 IF(B) 3, 8, 3
    3 DISC = B * B - 4. * A * C
      IF (DISC) 10, 7, 5
    5 N = 2
      X1 = (-B+SQRTF(DISC))/(2. * A)
      X2 = (-B-SQRTF(DISC))/(2. * A)
      GO TO 14
    7 N = 1
      X1 = -B/A
      GO TO 14
   10 N = 3
      GO TO 14
    8 IF(C/A) 9, 9, 10
    9 N = 2
      X1 = SQRTF(-C/A)
      X2 = -X1
      GO TO 14
   11 IF(B) 12, 13, 12
   12 N = 1
      X1 = -C/B
      GO TO 14
   13 N = 0
   14 RETURN
      END
```

2(b).

```
C       A PROGRAM FOR ROOTS OF QUADRATIC EQUATIONS
    1   READ 20, A, B, C
    2   PRINT 20, A, B, C
    3   CALL ROOTS (A,B,C,N,X1,X2)
    4   IF(K - 1) 5, 6, 11
   11   IF(K - 2) 99, 7, 8
    5   PRINT 21
        GO TO 1
    6   PRINT 22, X1
        GO TO 1
    7   PRINT 23, X1, X2
        GO TO 1
    8   PRINT 24
        GO TO 1
   99   STOP
   20   FORMAT(3F16.8)
   21   FORMAT(24H□NO□INTERESTING□□SOLUTION)
   22   FORMAT(17H□ONE□□SOLUTION□X=□,F16.8)
   23   FORMAT(19H□TWO□SOLUTIONS□X1=□,F16.8,5H□X2=□,F16.8)
   24   FORMAT(22H□SOLUTIONS□ARE□COMPLEX)
        END
```

3.

```
        SUBROUTINE SUBF(X,Y,T,K)
        IF(X - 2)2, 4, 2
    4   RETURN
    2   K = 1
        T = ((X ** 3 + Y) ** 2 + 5.)/(X - 2.)
        RETURN
        END
```

```
        CALL SUBF(R, S, Z, KTEST)
        IF(KTEST) 13, 12, 13
   12   PRINT 50
        GO TO 60
   50   FORMAT(21H□Z□CANNOT□BE□COMPUTED)
   13   Z = Z + 6. * T
```

Answers to Exercises P5-6

1.

```
      SUBROUTINE CCWTCH(N, IS, IC, ICOUNT)
      DIMENSION IS(100)
      ICOUNT = 0
      M = 1
    2 CALL CHEKCH (N, IS, M, IC, LOC)
      IF(LOC) 4, 6, 4
    4 ICOUNT = ICOUNT + 1
      M = LOC + 1
      GO TO 2
    6 RETURN
      END
```

2. The symbols) and (must be input for the main program and be included in the subroutine parameter list, say call them NPAR1 and NPAR2, respectively. Also let ERROR equal 0 for a correctly written expression, 1 for a negative counter value and 2 for a nonzero counter value at the end of the scan.

```
      SUBROUTINE PAREN (N, IS, NPAR1, NPAR2, ERROR)
      DIMENSION IS(100)
      ICOUNT = 0
      DO 50 I = 1, N
      IF(IS(I) - NPAR1) 7, 4, 7
    4 ICOUNT = ICOUNT - 1
      IF(ICOUNT) 6, 50, 50
    6 ERROR = 1
      RETURN
    7 IF(IS(I) - NPAR2) 50, 8, 50
    8 ICOUNT = ICOUNT + 1
   50 CONTINUE
      IF(ICOUNT) 11, 10, 11
   10 ERROR = 0
      RETURN
   11 ERROR = 2
      RETURN
      END
```

3.

```
      SUBROUTINE CONTST(N, IS, K, IC, ICOUNT)
      DIMENSION IS(100), IC(100)
      ICOUNT = 0
      M = 1
    2 CALL CHEKST (N, IS, M, K, IC, LOC)
      IF(LOC) 6, 6, 4
    4 ICOUNT = ICOUNT + 1
      M = LOC + 1
      GO TO 2
    6 RETURN
      END
```

4(a)

```
      SUBROUTINE AVER(M,N,IA,V)
      DIMENSION IA(1000)
      NUM = N - M + 1
      ISUM = 0
      IF(NUM) 3, 6, 3
    3 DO 4 I = M, N
      ISUM = ISUM + IA(I)
    4 CONTINUE
      V = ISUM/NUM
      RETURN
    6 V = -50
      RETURN
      END
```

4(b)

```
      DIMENSION IA(1000)
      READ 11,K,(IA(I),I = 1,K)
    1 READ 12,M,N
      CALL AVER(M,N,IA,AVERAGE)
      PRINT 12,M,N,AVERAGE
      GO TO 1
   11 FORMAT(I5(14I5))
   12 FORMAT(2I10,F10.2)
      END
C     DEFINITION OF AVER GOES HERE
      END
```

Chapter TF7

SOME MATHEMATICAL APPLICATIONS

The material here will parallel closely that in the student's Chapter F7. Since no new FORTRAN concepts have been introduced, discussions are limited to specific points regarding the exercises.

1.

```
      PRINT 1
    1 FORMAT(5X, 1HA, 9X, 1HB, 7X, 4HEPSI, 15X, 4HROOT)
      CALL ZEROK(0., 2., .1, 1)
      CALL ZEROK(.1, 1., .15,2)
      CALL ZEROK(0., 2., .4, 3)
      CALL ZEROK(0., 2., .1, 4)
      CALL ZEROK(0.4, 4., .1, 5)
      CALL ZEROK(C., 2., 1.E-4, 3)
      STOP
      END
C          SUBROUTINE ZEROK GOES HERE
C     THE FUNCTION SUBPROGRAM CALLED FUNCT
C     IS GIVEN HERE
      FUNCTION FUNCT (X,K)
      IF (K-2) 1, 2, 33
   33 IF (K-4) 3, 4, 5
    1 FUNCT = (X * X - 1.) * X - 1.
      RETURN
    2 FUNCT = X + LOGF(X)
      RETURN
    3 FUNCT = 5. - X - S. * SINF(X)
      RETURN
    4 FUNCT = (X * X - 3.) * X - 2.
      RETURN
    5 FUNCT = ((X + 2.) * X - 13.) * X + 10.
      RETURN
      END
```

Calculated results:

(1.) [illegible] 1.3438, EPSI = 0.1

(2.) 0.60625, EPSI = 0.15

(3.) 0.87500, EPSI = 0.4; 0.94565, EPSI = 10^{-4}

(4.) (Not available this edition)

(5.) Method is inapplicable.

TF7

2.

```
      DO  1   I=1,41
      X= I
      X= X/2. -10.5
      Y1= F11(X)
      Y2= F12(X)
      Y3= F1(X)
      Y4= F2(X)
      Y5= F3(X)
      Y6= F13(X)
      Y7= F4(X)
      Y8= F5(X)
      PRINT 2,X,Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8
    2 FORMAT( 9F10.2)
      STOP
      END
      FUNCTION  F11(X)
      F11= (X**2 -2.)*X -5.
      RETURN
      END
      FUNCTION F12(X)
      F12 = ((X+3.)*X-2.)*X-4.
      RETURN
      END
      FUNCTION F1(X)
      F1= ((3.*X-2.)*X**2+7.)*X-4.
      RETURN
      END
      FUNCTION F2(X)
      F2= (X**2-1.)*X-1.
      RETURN
      END
      FUNCTION F3(X)
      F3= (X-3.)*X-4.*(SINF(X))**2
      RETURN
      END
      FUNCTION F13(X)
      F13= X-SINF(X)/COSF(X)
      RETURN
      END
      FUNCTION F4(X)
      F4= X+LOGF(X)
      RETURN
      END
      FUNCTION F5(X)
      F5= 5.-X-5.*SINF(X)
      RETURN
      END
      END
```

3. (a) 15.03 (or 15 to the nearest foot)

(b) 15.65 feet

4.
```
C       MAIN
        CALL ZEROKL(1, LABEL, 0., 3.14159/2., 1.E-4,A)
        IF (LABEL) 4, 2, 4
      2 CALL ZEROKL (2, LABEL, 0., 3.14159/2., 1.E-4,B)
        IF (LABEL) 4, 3, 4
      3 PRINT 101, A, B
    101 FORMAT (17H□ROOT□OF□F(X)□IS□,E20.8,
     1          17H□ROOT□OF□G(X)□IS□,G20.8)
        STOP
      4 PRINT 102
    102 FORMAT (27H□METHOD□IS□INAPPLICABLE□FOR,
     1          15H□F(X),□OR,□G(X)  )
        STOP
        END
C            SUBROUTINE ZEROKL GOES HERE
C            FUNCTION DEFINITION FOLLOWS
        FUNCTION FUNCT(X,K)
        IF (K-1) 1, 1, 2
      1 FUNCT = SINF(X) - .66666667 * X
        RETURN
      2 FUNCT = SINF(X)/COSF(X) - .1 * X
        RETURN
        END
```

5.
```
C       MAIN
        CALL ZEROKL(1, LABEL, 0., .5, 1.E-4, F1)
        CALL ZEROKL(2, LABEL, .5, 1., 1.E-4, F2)
        CALL ZEROKL(3, LABEL, .707, 1, 1.E-4, RG2)
        CALL ZEROKL(4, LABEL, RG2, 1., 1.E-4, RG3)
        CALL ZEROKL(5, LABEL, RG3, 1., 1.E-4, RG4)
        CALL ZEROKL(6, LABEL, RG4, 1., 1.E-4, RG5)
        IF (LABEL) 7, 8, 7
      8 PRINT 101
    101 FORMAT (5H□OOPS)
        STOP
      7 PRINT 102, F1, F2, RG2, RG3, RG4, RG5
    102 FORMAT (2E20.8/4E20.8)
        STOP
        END
C       THE FUNCTION FUNCT(X,K)
        FUNCTION FUNCT(X,K)
        IF (K-2) 1, 2, 33
     33 IF (K-4) 3, 4, 53
     53 IF (K-6) 5, 6, 6
      1 FUNCT = X * SQRTF(1 - X * X) - .25
        RETURN
      2 FUNCT = SQRTF(1 - X * X) - X
        RETURN
      3 FUNCT = SQRTF(1 - X * X) - X ** 2
        RETURN
      4 FUNCT = SQRTF(1 - X * X) - X ** 3
        RETURN
      5 FUNCT = SQRTF(1 - X * X) - X ** 4
        RETURN
      6 FUNCT = SQRTF(1 - X * X) - X ** 5
        RETURN
        END
```

Answers to Exercises F7-2

1. (a) If EPSI is too small for the accuracy of the machine being used, it may not terminate.

   (b) If N is greater than 100, the storage set aside for T will be exceeded.

   (c) Replace statement 8 with

```
    8  IF (N - 100) 3, 7, 7
```

   The remainder of the program could be

```
    3  N = N - 1
       GO TO 2
    7  PRINT 103
  103  FORMAT (30H ERROR TOLERANCE EXCEEDED)
    9  AREA = T(N)
       PRINT 106, AREA
  106  FORMAT (8HAREA = , E20.8)
       STOP
       END
```

2. We may eliminate the calculation of 2^n for each n. Also, since only the current sum and the previous sum are needed, we may denote them by AREA and OLAREA, respectively, thus eliminating the use of subscripted variables.

```
      READ 1,EPSI
    1 FORMAT (E 20.8)
      OLAREA = 0.5 * (1.0 + 0.5)
      M = 1
      H = 1
      N = 1
    2 M = 2*M
      L = M-1
      H = H/2.
      S = 0
      DO 3 K = 1,L,2
      FK = K
    3 S = S+1.0/(1.0+FK*H)
      AREA = 0.5*OLAREA + H*S
      IF ( ABSF(AREA-OLAREA)-EPSI)4,5,5
    5 IF(N-100) 9,7,7
    9 N = N+1
      TZERO = AREA
      GO TO 2
    7 PRINT 8
    8 FORMAT (30H ERROR TOLERANCE NOT SATISFIED)
    4 PRINT 6,EPSI,T1
    6 FORMAT (2E20.8)
      STOP
      END
```

For EPSI = 0.01, T1 = 0.69412.

For EPSI = 0.001, T1 = 0.69339.

3. Let LIMIT be the maximum number of iterations to be performed.

```
    5  READ 1, LIMIT
    1  FORMAT (I10)
       PRINT 10
   10  FORMAT (12X,2HT1 17X,4HDIFF 16X,5HLIMIT)
       TZERO = 0.5 * (1.0 + 0.5)
       N = 1
       M = 1
       H = 1.
    2  M = 2 * M
       L = M - 1
       H = H/2.
       S = 0
       DO 3 K = 1, L, 2
       FK = K
    3  S = S + 1./(1.+FK* H)
       T1 = 0.5 * TZERO + H * S
       DIFF = T1 - TZERO
       IF (N - LIMIT) 9, 4, 4
    9  N = N + 1
       TZERO = T1
       GO TO 2
    4  PRINT 6, T1, DIFF, LIMIT
    6  FORMAT(2E20.8,I15)
       STOP
       END
```

For LIMIT = 15, T1 = 0.69314.

4. Before the END statement we may insert GO TO 8.

5.

```
       PRINT 10
   10  FORMAT (10X,4HAREA 16X,1HN)
    4  READ 1, N
    1  FORMAT (I10)
       S = (1. + 1./2.)/2.
       KLAST = N - 1
       XN = N
       DO 2 K = 1, KLAST
       XK = K
    2  S = S + 1./(1. + XK/XN)
       AREA = S/XN
       PRINT 3, AREA, N
    3  FORMAT(E20.8,I15)
       GO TO 4
       END
```

For N = 5, AREA = 0.69563
For N = 25, AREA = 0.69325
For N = 75, AREA = 0.69316
For N = 125, AREA = 0.69315
For N = 200, AREA = 0.69315

TF7-3 Area under curve: the general case

Answers to Exercises F7-3

1. (a)

```
      .14159
     Z=AREA2(0.,PI,5000)
     PRINT 4,Z
   4 FORMAT(1XE20.8)
     STOP
     END
     FUNCTION AREA2(A,B,N)
     FN=N
     H=(B-A)/FN
     S= (FUNCT(B)+FUNCT(A))/2.
     L= N-1
     IF (L) 7,2,3
   3 DO 1 K=1,L
     FK= K
   1 S = S+FUNCT(A+FK*H)
   2 AREA2= H*S
     RETURN
     END
     FUNCTION FUNCT(X)
     FUNCT= SINF(X)
     RETURN
     END
     END
```

The area under the sine curve is 2.000, while for a semicircle of diameter π the area is 3.876.

(b) Of course this is not the usual method for printing a logarithm table. The problem was given to connect again the area method to the introductory discussion of ℓnx. The correct values for ℓnx are these:

x	ℓnx	x	ℓnx
1	0.00000	31	3.43399
6	1.79176	36	3.58352
11	2.39790	41	3.71357
16	2.77259	46	3.82864
21	3.04452	51	3.93183
26	3.25810		

2. The variable M gives the current number of subdivisions. These revisions would accomplish the desired termination.

(1) N, the maximum number of subdivisions, may be included as an argument of the function: AREA (A, B, EPSI, N).

(2) The remainder of the program after Statement 5 could be

```
      IF(M - N) 6, 7, 7
    6 GO TO 1
    7 PRINT 8, T1
    8 FORMAT (7H□AREA□= E20.8,30H□□□ACCURACY□CRITERION□EXCEEDED)
    2 AREA = T1
      RETURN
      END
```

3. (a)

```
      DO 4  L=1,3
      N =  2**(L-1)
      Z=AREA2(1.,3.,N)
    4 PRINT 3,N,Z
    3 FORMAT(4H□N=□I3, 9H□□□AREA=□E20.8)
      Z=AREA(1.,3.,0.001)
      PRINT 5,Z
    5 FORMAT(6H□AREA=E20.8)
      STOP
      END
C   FUNCTION AREA2 GOES HERE
C   FUNCTION AREA GOES HERE
      FUNCTION FUNCT(X)
      FUNCT=.43429/X
      RETURN
      END
      END
```

Calculated results:

Subdivisions	EPSI	Area
1		0.57905
2		0.50667
4		0.48496
	10^{-3}	0.47724

3. (b)

```
      DO 6 L2 = 1, 3
      N = 2**(L2 - 1)
      Z = AREA2(-2., 2., N)
    6 PRINT 3, N, Z
    3 FORMAT(4H N = I3, 9H   AREA = E20.8)
      Z = AREA(-2., 2., 0.001)
      PRINT 5,Z
    5 FORMAT(6H AREA = E20.8)
      STOP
      END
C     FUNCTION AREA2 GOES HERE
C     FUNCTION AREA GOES HERE
      FUNCTION FUNCT(X)
      FUNCT = (3.*X + 2.) * X + 1.
      RETURN
      END
      END
```

Calculated results:

Subdivisions	EPSI	Area
1		52.00
2		28.000
4		22.000
	10^{-3}	20.000

(c)

```
      DO 7  L3=1,3
      N = 2**(L3-1)
      Z=AREA2(1.,4.,N)
    7 PRINT 3,N,Z
    3 FORMAT(4H N= I3, 9H    AREA= E20.8)
      Z=AREA(1.,4.,0.001)
      PRINT 5,Z
    5 FORMAT(6H□AREA=E20.8)
      STOP
      END
C   FUNCTION AREA GOES HERE
C   FUNCTION AREA2 GOES HERE
      FUNCTION FUNCT(X)
      FUNCT=(X-1.)*X*X
      RETURN
      END
      END
```

Calculated results:

Subdivisions	EPSI	Area
1		72.00
2		50.063
4		44.578
	10^{-3}	42.750

Comment

After their experience with ZEROK, some students may introduce an index K into the parameter list of AREA and AREA2 in order to avoid "packaging" the two subroutines with each function separately.

4.
```
C
C     MAIN
      T = AREA(0., 2., 1.E-4)
      PRINT 200, T
  200 FORMAT (8H□AREA□=□,E20.8)
      STOP
      END
C     FUNCT IS DEFINED HERE
      FUNCTION FUNCT(X)
      FUNCT = SQRTF(4. - X * X)
      RETURN
      END
```

TF7-4 Simultaneous linear equations: developing a systematic method of solution

Answers to Exercises F7-4

1.

```
      3  READ 1, A11, A12, B1, A21, A22, B2
         PRINT 1, A11, A12, B1, A21, A22, B2
      1  FORMAT (1XF9.0,5F10.0)
         AA11 = A11/A11
         AA12  = A12/A11
         BB1 = B1/A11
         AA21 = A21 - A21 * AA11
         AA22 = A22 - A21 * AA12
         BB2 = B2 - A21 * BB1
         X2 = BB2/AA22
         X1 = BB1 - AA12 * X2
         PRINT 2, X1, X2
      2  FORMAT(7H□□□X7□=E20.8, 5X4HX2□= E20.8)
         GO TO 3
         END
```

2.

(a)	$x = 1.6250$	$y = 0.75000$
(b)	$x = 1.8636$	$y = -0.81818$
(c)	$x = 2.4706$	$y = -1.1471$
(d)	$x = 1.5000$	$y = -2.5000$
(e)	$x = 0.65217$	$y = -1.2609$
(f)	$x_1 = 0.18958$	$x_2 = -0.33977$
(g)	$x_1 = 2.1933$	$x_2 = -0.30269$

Although this program will solve our problems, we have actually computed much more than necessary. The next section of the text shows how to solve equations more efficiently.

TF7-5 Simultaneous linear equations: Gauss algorithm

Answers to Exercises F7-5 Set A

1.

```
      DO 100 J = 3,3
      A(2,J) = A(2,J)/A(2,2)
100   CONTINUE
      B(2) = B(2)/A(2,2)
```

2.

```
      DO 100 K = 1,2
      J1 = K + 1
      DO 200 J = J1,3
      A(K,J) = A(K,J)/A(K,K)
200   CONTINUE
      B(K) = B(K)/A(K,K)
100   CONTINUE
```

3.

```
      IF (K - 3) 5, 60, 60
5     KP1 = K + 1
      DO 500 I = KP1,3
      DO 600 J = KP1,3
      A(I,J) = A(I,J) - A(I,K) * A(K,J)
600   CONTINUE
      B(I) = B(I) - A(I,K) * B(K)
500   CONTINUE
60    ~~~~~
```

4.

```
      DO 100 K = 1,3
      KP1 = K + 1
      DO 200 J = KP1,3
         A(K,J) = A(K,J)/A(K,K)
200   CONTINUE
      B(K) = B(K)/A(K,K)
      IF (KP1-4) 5, 60, 60
5     DO 500 I = KP1, 3
         DO 600 J = KP1, 3
         A(I,J) = A(I,J) - A(I,K) * A(K,J)
500   CONTINUE
      B(I) = B(I) - A(I,K) * B(K)
600   CONTINUE
60    ~~~~~
```

Answers to Exercises F7-5 Set B

1.

```
      |DIMENSION A(3,3), B(3), X(3)
      |DO 150 I = 1,3
      |READ 31, (A(I,J), J = 1,3), B(I)
      |PRINT 31, (A(I,J), J = 1,3), B(I)
   31 |FORMAT (4F10.5)
  150 |CONTINUE
      |DO 100 K = 1,3
      |   KP1 = K + 1
      |   IF (K - 3) 2, 4, 4
    2 |DO 200 J = K1,3
      |   A(K,J) = A(K,J)/A(K,K)
  200 |   CONTINUE
    4 |B(K) = B(K)/A(K,K)
      |   IF (K - 3) 5, 100, 100
    5 |DO 500 I = KP1,3
      |DO 600 J = KP1,3
      |   A(I,J) = A(I,J) - A(I,K) * A(K,J)
  600 |   CONTINUE
      |   B(I) = B(I) - A(I,K) * B(K)
  500 |   CONTINUE
  100 |   CONTINUE
      |DO 900 I = 1,3
      |   I1 = 4 - I
      |   X(I1) = B(I1)
      |   IF (I - 1) 11, 900, 11
      |   I1M1 = I1 - 1
      |   DO 1100 J = 1, I1M1
      |      J1 = 4 - J
      |   X(I1) = X(I1) - A(I1,J1) * X(J1)
 1100 |   CONTINUE
  900 |   CONTINUE
      |PRINT 150, (X(I), I = 1,3
      |STOP
      |END
```

2. (a) $x_1 = -10.000$
$x_2 = 1.8824$
$x_3 = 15.471$

(b) $x_1 = 2.6279$
$x_2 = -0.23256$
$x_3 = -1.8372$

(c) $x_1 = 1.2289$
$x_2 = 0.20482$
$x_3 = -0.83133$

(d) $x_1 = 2.8154$
$x_2 = 1.7077$
$x_3 = -0.15385$

3. (a) $x_1 = -3.0619$
$x_2 = 5.9268$
$x_3 = 0.13861$

(b) $x_1 = 0.66311$
$x_2 = 5.1741$
$x_3 = -1.5221$

Answers to Exercise F7-5 Set C

```
      SUBROUTINE GAUSS(N,A,B,X)
      DIMENSION A(15,15), B(15), X(15)
      DO 100 K = 1,N
         KP1 = K + 1
         IF (K - N) 2, 4, 4
    2 DO 200 J = K1,N
         A(K,J) = A(K,J)/A(K,K)
  200    CONTINUE
    4 B(K) = B(K)/A(K,K)
         IF (K - N) 5, 100, 100
    5 DO 500 I = KP1,N
      DO 600 J = KP1,N
         A(I,J) = A(I,J) - A(I,K) * A(K,J)
  600    CONTINUE
         B(I) = B(I) - A(I,K) * B(K)
  500    CONTINUE
  100    CONTINUE
      DO 900 I = 1,N
         I1 = N + 1 - I
         X(I1) = B(I1)
         IF (I-1) 11, 900, 11
         I1M1 = I1-1
         DO 1100 J = 1, I1M1
            J1 = N + 1 - J
         X(I1) = X(I1) - A(I1,J1) * X(J1)
 1100    CONTINUE
  900    CONTINUE
      RETURN
      END
```

Here is a sample calling program.

```
      DIMENSION A(15,15), B(15), X(15)
      READ 100,N
  100 FORMAT (I5)
      DO 150 I = 1,N
         READ 101, (A(I,J), J = 1,N)
         PRINT 101, (A(I,J), J = 1,N)
  101    FORMAT (5F10.5)
  150 CONTINUE
      READ 101 (B(I), I = 1,N)
      PRINT 101 (B(I), I = 1,N)
      CALL GAUSS (N,A,B,X)
      PRINT 101, (X(I), I = 1,N)
      STOP
      END
```

Answers to Exercises F7-5 Set D

1. (a) We add the index L as fifth argument for indicating an alternate exit in the event the matrix A is singular.

(b) Immediately after the statement

```
KP1 = K + 1
```

we now insert the following statements:

```
      |TMAX = ABSF(A(K,K))
      |M = K
      |DO 1400 I = KP1,N
      |   IF ABSF(A(I,K) - TMAX)1400,16,16
      |   TMAX = ABSF(A(I,K))
      |   M = I
 1400 |   CONTINUE
      |IF (TMAX) 18,22,18
   22 |L = 1
      |RETURN
   18 |IF (M-K) 19,2,19
      |DO 1900 J = K,N
      |   COPY = A(K,J)
      |   A(K,J) = A(M,J)
      |   A(M,J) = COPY
 1900 |   CONTINUE
      |   COPY = B(K)
      |   B(K) = B(M)
      |   B(M) = COPY
- - - + - - - - - - - - - -
    2 |
```

2. Results with partial pivoting

(a) $x_1 = 1.1805$
$x_2 = -0.54135$
$x_3 = 0.59398$

(b) $x_1 = 10.550$
$x_2 = 3.9000$
$x_3 = -0.60000$

Without partial pivoting

(a) $x_1 = 0$
$x_2 = 0.092593$
$x_3 = 1.7963$

(b) $x_1 = 2.75$
$x_2 = 0$
$x_3 = 3.75$

Note: Of course the answers without pivot are wrong since the machine continued the calculation after the attempted division by zero. Some compilers may perform an error stop if division by zero is encountered.